STUDIES IN LEARNING CONTROL SYSTEMS

A Thesis Submitted
in partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

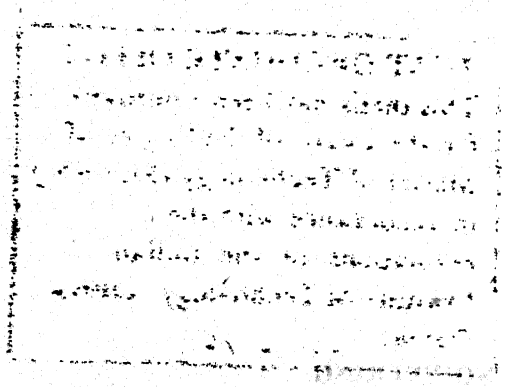
SHRIKANT H. AGARWAL

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JUNE, 1976

EE-1976-M-AGA-STU



CERTIFICATE

This is to certify that this work on, "Studies in Learning Control Systems", has been carried out under my supervision and has not been submitted elsewhere for a degree.

(R. Subramanian)
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur.

ACKNOWLEDGEMENTS

No work is done in isolation, and this one is no exception. Many people have influenced me, some through personal contacts and others, little indirectly through their work.

To my guide Dr. R. Subramanian, I am deeply indebted for his encouraging, sympathetic and understanding attitude throughout the period of investigation.

I extend my sincere thanks to my friends for many stimulating and instructive discussions.

Last but not the least, thanks are also due Mr. R.N. Srivastava who typed the thesis with extreme care.

S Agarwal

Shrikant H. Agarwal

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

A	State coefficient matrix
B	Control coefficient matrix
$\underline{b}$	Control coefficient vector
$\underline{\Psi}(t)$	Solution of the adjoint state equation
e	Napier's constant
H	Hamiltonian
J	Performance measure
λ	Relaxation coefficient in relaxation algorithm
$\|\ \|$	Euclidean norm
η	Error correction coefficient in error correction learning algorithm
P_S	A priori probability
Q	Code matrix
R	Response of an Adaline
$\underline{S}$	A pattern vector or stimulus
s_i	An element of a stimulus
Sgn	Signum function
z	A dummy variable
$\underline{U}(t)$	Control vector
$u(t)$	Scalar control input
$\underline{W}$	A weight vector
$\underline{w}_i$	An element of a weight vector
$\underline{X}(t)$	State vector

ABSTRACT

Artificial intelligence techniques have been applied to the design of control systems. In the present work a synthesis technique known as off-line (learning) training is considered, which achieves a practical realization of closed loop time optimal, fuel optimal, or minimum energy control laws. In this technique a training set in state space is obtained. A training set is a set of quantum regions of the state space for which the optimal control is known. A code is assigned to all such possible regions in order to identify a particular zone. The training set is used to train a learning controller which synthesizes the complete control law during the training by means of a learning algorithm. The function of a trained controller is to provide the optimal control for all points in state space with a limited training set.

In the present work the experimental results of the performance of the learning controller using different codes are obtained. A learning algorithm called the "relaxation algorithm" is also considered in order to investigate the learning characteristics of a learning controller. Experimental results are given for the same.

The application of a trainable controller is finally considered in a third order time optimal control system. The training set is obtained by generating optimal trajectories in the state space using the Neustadt's method and the quantum zones are identified for the training set through which the optimal trajectories pass. The experimental results are provided, demonstrating the performance and reliability of a trained controller using different codes.

All computer programs were written in FORTRAN IV Language to be run on IBM 7044 computer.

CHAPTER 1

THE OPTIMAL CONTROL PROBLEM

1.1 Introduction:

All physical processes described by a cause-effect relationship can be viewed as a system with inputs and outputs.

Let us assume that we are given a system, or, a "black box", to which we can apply certain input signals, in order to observe and measure the resultant output signals. Our ultimate objective is the determination of an input which will produce an output with certain desired characteristics, and which will in doing so, minimize the "cost" of operation (the term cost could be the time required fuel consumed or energy consumed or other indices of interest). We might try to achieve our objective by a trial and error procedure, by trying first one input and then another. But in general, this procedure will not guarantee a solution, besides being time consuming. However, we might look upon our efforts as experiments which could lead us to a description of the behaviour of the system and could indicate to us the particular output resulting from the application of a particular input. We generally attempt to solve these problems by

developing mathematical models in which equations relate the output to the input.

We shall consider systems which can be described by ordinary differential equations. We call such systems as Dynamical Systems. These can be represented mathematically as

$$\left.\begin{aligned}
\dot{x}_1 &= f_1(x_1, x_2 \ldots x_n;\ u_1, u_2 \ldots u_r) \\
\dot{x}_2 &= f_2(x_1, x_2 \ldots x_n;\ u_1, u_2 \ldots u_r) \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
\dot{x}_n &= f_n(x_1, x_2 \ldots x_n;\ u_1, u_2 \ldots u_r)
\end{aligned}\right\} \quad (1.1)$$

where $x_1, x_2 \ldots x_n$ are the n state variables of the system and $u_1, u_2 \ldots u_r$ are r input variables. The dot notation represents the time derivative.

If the system of above differential equation is represented in vector form then we have -

$$\dot{\underline{X}}(t) = \underline{F}(\underline{X}(t), \underline{U}(t), t) \quad (1.2)$$

If the dynamical system under consideration is linear, then (1.2) takes the form as -

$$\dot{\underline{X}}(t) = A(t)\,\underline{X}(t) + B(t)\,\underline{U}(t) \quad (1.3)$$

1.2 The Optimal Control Problem:

In all control problems we are interested in transferring the state of the system to a particular state from some initial state by application of proper control inputs. If the system is controllable then this can be achieved by applying suitable control inputs.

In order to evaluate the performance of a dynamical system quantitatively, we select a measure of performance. An optimal control is defined as one that minimizes (or maximizes) the performance measure. The selection of performance measure depends on the nature of problem at hand. For example, if the problem is to "move" the system from an initial state $\underline{X}(t_o)$ to a final state $\underline{X}(t_f)$ in minimum time, this is known as a time optimal control problem, where the control input has to be so chosen so as to minimize the performance measure which is the integral of time or -

$$J = \int_{t_o}^{t_f} dt \quad , \tag{1.4}$$

J being the performance measure.

Similarly, in some other problem we may be required to "move" the system from an initial state $\underline{X}(t_o)$ to a final state $\underline{X}(t_f)$ such that the fuel consumption is minimum, or in other words, if the control input $\underline{U}(t)$ represents a

variable proportional to the rate of fuel consumption, then the problem is to minimize J, where -

$$J = \alpha \int_{t_o}^{t_f} \sum_{i=1}^{r} u_i(t) \, dt \tag{1.5}$$

α being a constant of proportionality.

The general linear optimal control problem is to find an admissible control $\underline{U}^*(t)$ which causes the dynamical system

$$\dot{\underline{X}}(t) = A(t)\,\underline{X}(t) + B(t)\,\underline{U}(t)$$

to follow an admissible trajectory (a trajectory satisfying some constraints on state variables.) that minimizes the performance measure

$$J = h(\underline{X}(t_f), t_f) + \int_{t_o}^{t_f} g(\underline{X}(t), \underline{U}(t), t) \, dt \tag{1.6}$$

The equation (1.6) is a general form of the performance measure and $X(t_f)$ and t_f are the terminal state vector and the terminal time. In the present work we shall consider the performance measures dealing with the minimum time, fuel and energy control problems.

Let us consider the example of a spacecraft entering the earth's atmosphere. During the atmospheric re-entry, there exists an optimum entry angle that guarantees a safe

landing. If this re-entry angle is too large, the spacecraft may be bounced by the outer layers of atmosphere back to outer space, and if it is too low, then the spacecraft may get over-heated and burnt due to air friction. Another effect noted is over-heating of the spacecraft body due to high velocity attained with incorrect re-entry angle. The gas in the boundary layer between the shock-wave front and the spacecraft surface becomes ionised. A high temperature plasma is thus formed around the space vehicle, which may fully envelope it and the high frequency radio communication units it carries. This envelope of plasma can completely cut off the radio communication of the spacecraft with the ground station and thus re-entry can not be program controlled by ground station.

If during re-entry, the entry angle is disturbed beyond the safe limits, it is essential that this error be corrected in minimum possible time considering the great speed of spacecraft descent. Here the problem is to choose the proper control inputs to correct the error in minimum time.

Another example is a fuel optimal control system. In case of moon landing modules, the problem of minimum expenditure of fuel is very important. Let us assume that

a spacecraft is in the terminal descent phase of a lunar mission, and is descending vertically. The problem is to properly switch on or switch off the retro-motors in such a way that the module attains zero terminal velocity at some specified altitude with a minimum expenditure of fuel. Once the terminal condition is achieved the vehicle can hover by applying a thrust acceleration of one lunar g (acceleration due to gravity). Such a mission permits the inspection of the moon's surface and a suitable landing place can be selected.

In practical situations, such as the spacecraft problem, there are some constraints on the control inputs, for example the maximum thrust offered by the motors is always limited. In such cases we have the constrained input optimal control problem.

1.3 Theoretical Methods in Optimal Control:

From an historical point of view, the need for theory in the area of optimization arose, to a large degree, from the stringent requirements of aerospace systems. Such systems are inherently nonlinear and they are subject to constraints, i.e. limited thrust capability, acceleration constraints, fuel limitations, limited fin steering availability etc. Before the development of modern control

theory, the only body of theory available for control was that dealing with the analysis and design of servomechanisms. However, the theory of servomechanisms was not adequate for the analysis and design of unconstrained single-input-output, linear and time invariant systems. Moreover, most of these methods were focused upon the relative stability of the closed loop systems, the design tools (e.g. root locus, frequency response, describing function etc.) were graphical and qualitative in nature, and the design process was mostly done in a trial and error fashion, under the assumption of zero initial conditions. The Euler's equation method using the calculus of variation was applied to control problems, but it did not prove very useful for problems with constrained inputs.

The two main theoretical approaches to the control optimization problem have been (i) Bellman's Dynamic Programming method [1] which is based on the principle of optimality and (ii) Pontryagin's Minimum Principle [2] which can be viewed as an extension and application of classical calculus of variation to the optimal control problem.

The Dynamic Programming method was originally developed for discrete time problems and later on it was applied to continuous time problems. In the later form it is often referred to as the Hamilton-Jacobi-Bellman theory.

Its major disadvantage lies in the large computer memory requirements ("The curse of dimensionality").

The Minimum Principle was originally developed for continuous time problems, and has been extended to discrete-time problems. Its major disadvantage is that it provides, in general, only local necessary conditions for optimality. Its computational requirements are not as severe as those associated with dynamic programming.

The minimum principle, however, proves to be a very powerful method in investigating the nature of the optimal controls.

1.4 Application of Pontryagin's Minimum Principle to Investigate the Nature of Controls for Minimum Time, Fuel and Energy Problems:

As we have stated earlier, the minimum principle is a very powerful method to obtain a qualitative idea about the nature of optimal control, we will illustrate its use in time, fuel and energy optimal control problems. A detailed discussion of the minimum principle can be found in Pontryagin [2], or Athans and Falb [4].

1.4.1 Time-optimal-control problem:

Consider a linear time invariant system

$$\dot{\underline{X}}(t) = A\,\underline{X}(t) + B\,\underline{U}(t) \qquad (1.7)$$

where $\underline{X}$ is an n-vector, and $\underline{U}$ an r-vector. A and B are constant matrices. The constraint on $\underline{U}$ is

$$|u_i(t)| \leq 1 , \qquad i = 1,2,\ldots,r \tag{1.8}$$

The solution of (1.7) is given by

$$\underline{X}(t) = e^{At} \cdot \underline{X}(0) + e^{At} \int_0^t e^{-A\tau} B \, \underline{U}(\tau) \, d\tau \tag{1.9}$$

If the problem is to drive the system to zero state in time t_f, then we have

$$\underline{X}(0) = - \int_0^{t_f} e^{-A\tau} B \, \underline{U}(\tau) \, d\tau \tag{1.10}$$

A control which satisfies (1.10), drives the system from its initial state $\underline{X}(0)$ to $\underline{0}$.

The adjoint eqyation of the system (1.7) is given by,

$$\dot{\underline{\psi}}(t) = - A^T \underline{\psi}(t) \tag{1.11}$$

where $\underline{\psi}(t)$ is an n-vector and A^T is the transpose of A. The solution of (1.11) is given by

$$\underline{\psi}(t) = e^{-A^T t} \underline{\psi}(0)$$

where $\underline{\psi}(0)$ is a constant.

The performance measure is given by

$$J = \int_0^{t_f} 1 \cdot dt \tag{1.12}$$

Let us define Hamiltonian as

$$H = \underline{\psi}^T \dot{\underline{X}} + 1 \tag{1.13}$$

If an optimal control $\underline{U}^*(t)$ exists, then according to minimum principle, $\underline{U}^*(t)$ must minimize the Hamiltonian for all t, $0 \leq t \leq t_f$ 2,3 . Thus

$$H(\underline{U}^*(t), \underline{\psi}^{*T}(t), \dot{\underline{X}}^*(t)) \leq H(\underline{U}(t), \underline{\psi}^{*T}(t), \dot{\underline{X}}^*(t)) \qquad (1.14)$$

or,

$$\underline{\psi}^{*T}(t)\left[A\,\underline{X}^*(t) + B\,\underline{U}^*(t)\right] \leq \underline{\psi}^{*T}(t)\left[A\,\underline{X}^*(t) + B\,\underline{U}^*(t)\right] \qquad (1.15)$$

Thus from the equation (1.15) we notice that if no element of $\underline{\psi}^{*T}(t).B$ vanishes for all t, $0 \leq t \leq t_f$; then the control that satisfies the equation (1.15) is given by

$$u_i^*(t) = -\,\mathrm{Sgn}\left[q_i\right], \qquad i = 1, 2, \ldots r \qquad (1.16)$$

where q_i are the elements of the vector

$$\underline{Q} = \underline{\Psi}^{*T}B$$

From the equation (1.16) the bistable nature of the optimal control is clear from Fig. 1.1.

1.4.2 Fuel-optimal-control problem:

The fuel optimal control problem has the following four forms.

(i) The fuel consumption is minimized without any restriction on the response time, the response time being

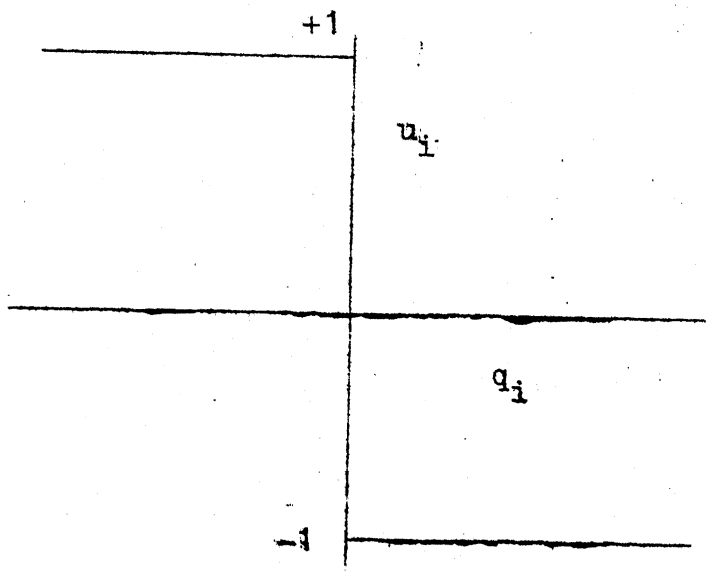


Fig. 1.1 Time-Optimal-Control

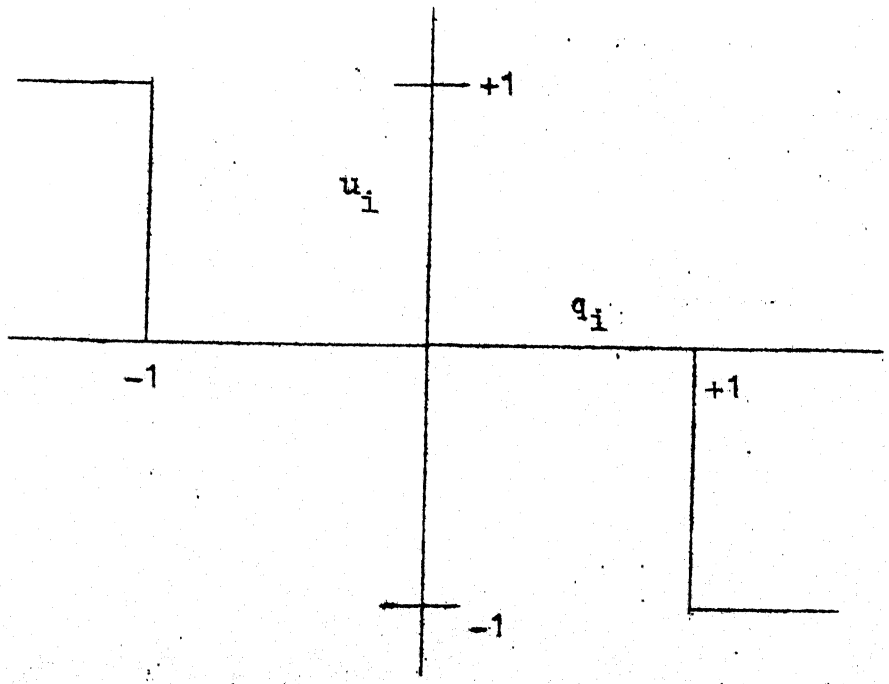


Fig. 1.2 Fuel-Optimal-Control

the time required to drive the system from any initial state $\underline{X}(0)$ to a final state $\underline{X}(t_f)$.

(ii) The fuel consumption is minimized with the response time fixed a priori.

(iii) The fuel consumption is minimized to drive the system from any initial state $\underline{X}(0)$ to a final state $\underline{X}(t_f)$ such that the response time t_f is a multiple of the minimum time t^*; t^* being the minimum time to drive the system from the same initial state to the same final state, without any constraint on the consumed fuel.

(iv) A linear combination of fuel and the response time is minimized.

We will restrict ourselves to problems where response time is not fixed a priori.

Consider the system (1.7)

$$\dot{\underline{X}}(t) = A\,\underline{X}(t) + B\,\underline{U}(t) \tag{1.7}$$

The performance measure to be minimized has the form,

$$J = \int_0^{t_f} \sum_{i=1}^{r} u_i(t)\, dt \tag{1.17}$$

Notice that $u_i(t)$ is a measure of the rate of fuel consumption.

The Hamiltonian is defined as,

$$H = \underline{\Psi}^T \dot{\underline{X}} + \sum_{i=1}^{r} |u_i(t)| \qquad (1.18$$

Substituting the expression for $\dot{\underline{X}}$ in (1.18), we have

$$H = \underline{\Psi}^T \left[A \underline{X} + B \underline{U} \right] + \sum_{i=1}^{r} |u_i(t)| \qquad (1.19$$

Let

$$\underline{Q} = \underline{\Psi}^T B = \left[q_1, q_2, \ldots q_r \right]^T \qquad (1.20)$$

Now according to the minimum principle, if there exists an optimal control $\underline{U}^*$, then it must minimize the Hamiltonian for all t, $0 \leq t \leq t_f$. In other words, $\underline{U}^*$ should minimize the expression,

$$E = q_i u_i + |u_i| , \qquad i = 1, 2, \ldots r \qquad (1.21)$$

We have,

$$E = q_i u_i + u_i = \begin{cases} \left[1 + q_i\right] u_i, & \text{if } u_i \text{ is positive} \\ \left[-1 + q_i\right] u_i, & \text{if } u_i \text{ is negative} \end{cases} \qquad (1.22)$$

Notice from (1.22) that if $q_i \geq 1$, then minimum value of E is obtained if $u_i = -\operatorname{Sgn} q_i$. Similarly, if $q_i \leq -1$ then minimum value of E is obtained if $u_i = -\operatorname{Sgn}\left[q_i\right]$.

Now if $-1 < q_i < +1$, then we find out that E is minimized if $u_i = 0$.

Thus the optimal control is given by,

$$\left.\begin{aligned} u_i &= -\operatorname{Sgn} q_i \ , \quad \text{if } |q_i| \geq 1 \\ u_i &= 0 \qquad\quad , \quad \text{if } -1 < q_i < +1 \\ &\text{for } i = 1, 2, \ldots r \end{aligned}\right\} \tag{1.23}$$

Thus the tristable nature of the optimal control is clear from Fig. 1.2.

1.4.3 Minimum-energy-control problem:

A class of problems exists where the square of the control signal is proportional to power, and its time integral is a measure of total energy used.

Consider the state equation (1.7)

$$\dot{\underline{X}}(t) = A\,\underline{X}(t) + B\,\underline{U}(t) \tag{1.7}$$

We have the performance measure

$$J = \frac{1}{2}\int_0^{t_f} \sum_{i=1}^{r} u_i^2(t)\, dt \tag{1.24}$$

We define the Hamiltonian as

$$H = \underline{\Psi}^T \dot{\underline{X}} + \frac{1}{2}\sum_{i=1}^{r} u_i^2(t) \tag{1.25}$$

or,

$$H = \underline{\psi}^T \left[A \underline{X} + B \underline{U} \right] + \frac{1}{2} \sum_{i=1}^{r} u_i^2(t) \qquad (1.26$$

We notice from equation (1.26) that the expression

$$E = \frac{1}{2} u_i(t) + q_i \; u_i(t)$$

where q_i is as defined in (1.20) will be minimized if $u_i(t)$ is so chosen that,

$$\left.\begin{aligned} u_i(t) &= -1 , \quad \text{if} \quad q_i \geq 1 \\ u_i(t) &= +1 , \quad \text{if} \quad q_i \leq -1 \\ \text{and,} \qquad u_i(t) &= -q_i , \quad \text{if} \quad -1 < q_i < +1 \end{aligned}\right\} \qquad (1.27)$$

Thus from Fig. 1.3, the nature of energy optimal control is clear.

1.5 Implementing the Results of the Optimal Control Theory:

In the previous section we have shown that the optimal control vector $\underline{U}^*(t)$ depends on the solution of the adjoint equation of the original state equation. The solution $\underline{\psi}^*(t)$, of the adjoint equation depends on an initial value, $\underline{\psi}^*(0)$ and this initial value, depends in some unknown manner on the initial state vector $\underline{X}(0)$. We should point out that upto

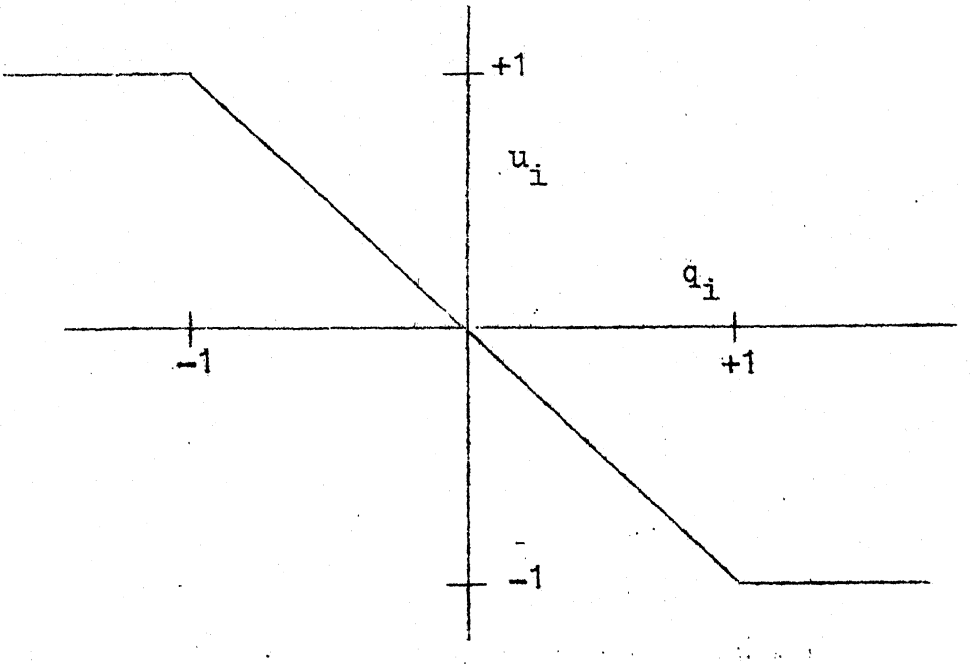


Fig. 1.3 Energy-Optimal-Control

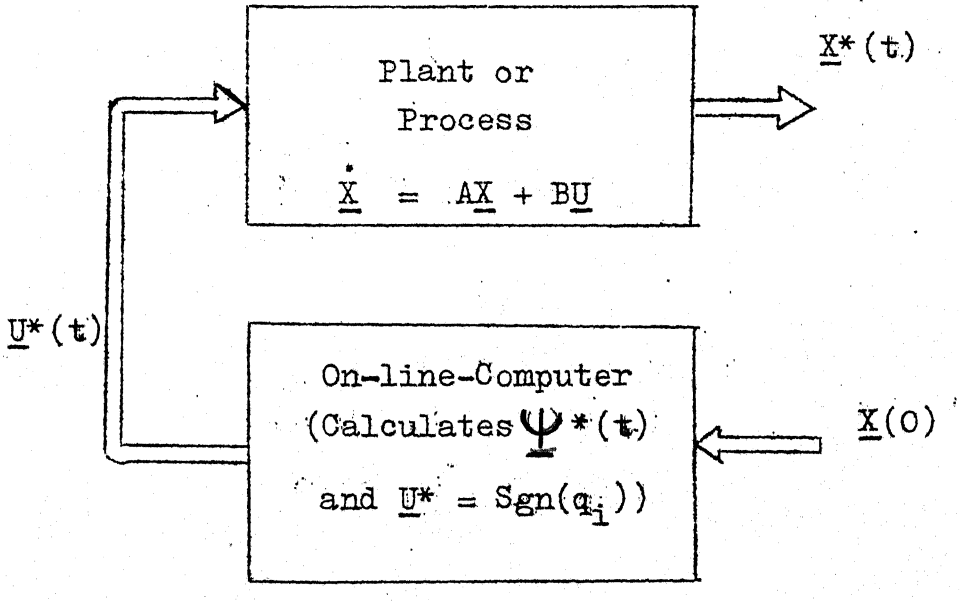


Fig. 1.4 Implementing the Optimal Control Laws by On-Line Computer

date there are no formulae available to relate $\underline{\psi}^*(0)$, to $\underline{X}(0)$. However, there exist a number of iterative methods to compute $\underline{\psi}^*(0)$ and the optimal controls. Plant [5] gives a good account of these methods. These methods usually start with an initial guess of $\underline{\psi}^*(0)$ and by successive approximation final $\underline{\psi}^*(0)$ is obtained. The method due to Neustadt [3] appears to be particularly attractive as it can be readily programmed on a digital computer.

One method that one can suggest to implement the optimal control laws, is to use an on line digital computer which can calculate $\underline{\psi}^*(0)$ by using some iterative algorithms [5], and generate the optimal control in real time. It will be noticed that such a method tends to be of an open loop type as time happens to be an independent variable. We know that a more reliable system will be obtained if the optimal control is generated as a function of the state variables, as it will constitute a closed loop system. Athans [4] points out that generally for systems of order higher than 3, the optimal control theory yields optimal controls as a function of time, and it is very difficult to find out a closed loop solution for controls as a function of state variables.

As mentioned earlier to implement the optimal control laws, an on line computing scheme can be used, but such a scheme may not be possible due to the following reasons.

(i) In applications such as re-entry of space-crafts, time can not be lost in mere computation, as time in such a case is very precious.

(ii) Due to limitation of space, it may not be possible to install an on line computer.

(iii) A reliable operation may not be guaranteed as optimal control generated will be function of time and not of state variables.

A general block diagram of this method is shown in Fig. 1.4.

Another simple method to implement the optimal control laws is to use analog computers to generate optimal controls [4],[6]. This method is useful only when the order of plant is small and optimal control is known as a function of state variables, with simple switching surfaces. We shall demonstrate this method for a simple problem of second order, and single input.

Let the system dynamics be

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

with the constraint

$$|u| \leq 1$$

Let the performance index be

$$J = \int_0^{t_f} 1 \cdot dt$$

Using the minimum principle and some manipulations have the closed form solution for the switching trajectory as,

$$x_1 = -\frac{1}{2} x_2 \, x_2$$

This equation describes a partitioning curve in the state space, with the states of same optimal control on one side and, the states of the opposite optimal control on the other. The optimal control is bang-bang type. Figure 1.5 depicts the switching curve with zones of constant control. The above optimal control law can be implemented using the analog computer elements as shown in the Fig. 1.6.

1.6 Conclusions:

It is clear from the above discussion that the present implementation schemes are inadequate in applications such as re-entry control of spacecrafts. It should be noted that the spacecraft re-entry physics is a subject, about which little is known to us. The outer layers of atmosphere have a random nature. In cases of re-entry in the other planets such as Mars, we know little about the dynamics of its atmospheric envelope. Thus we feel that in such cases

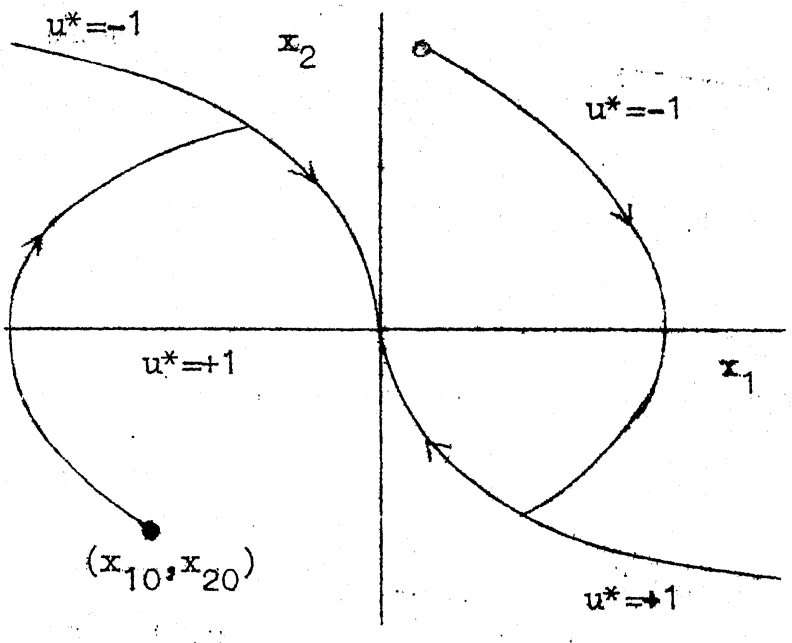


Fig. 1.5 Time optimal switching curve

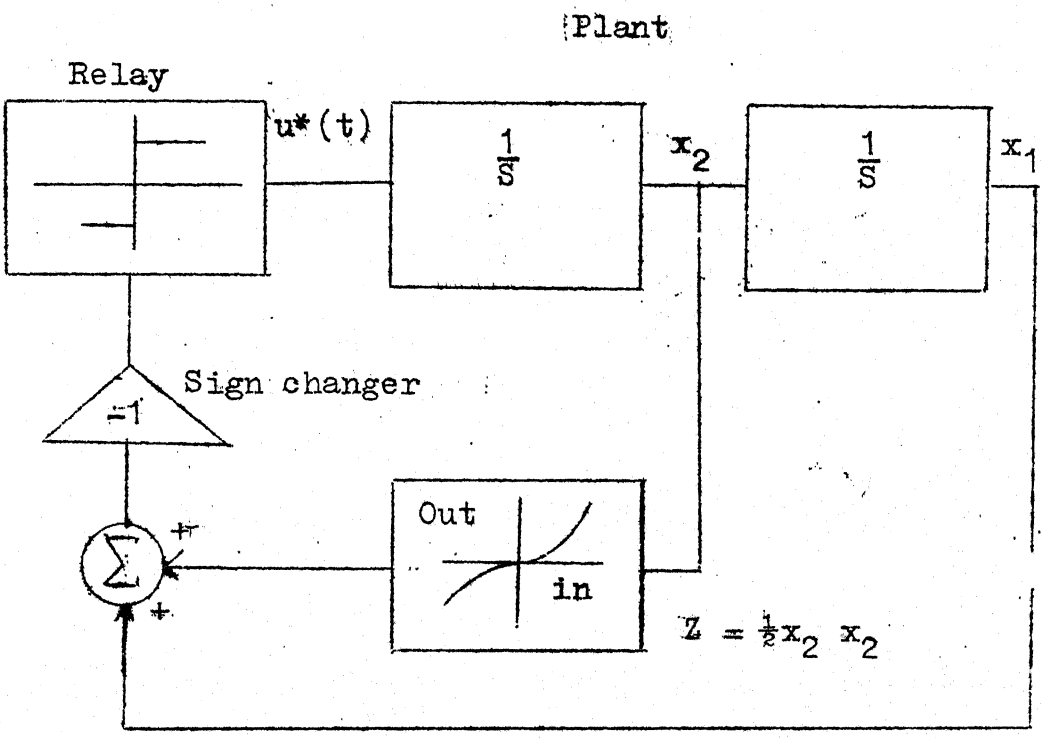


Fig. 1.6 Hardware implementation of time-optimal control law

there is a need of developing a control system technology to match these needs. The techniques of Artificial Intelligence find extensive use in control systems. In the present work the analysis and simulation of learning controllers is carried out.

CHAPTER 2

ARTIFICIAL INTELLIGENCE AND LEARNING MACHINES

2.1 Limitations of Optimal Control Law Implementation:

In Chapter 1 we mentioned the existing methods in optimal control theory, and in particular applied Pontryagin's minimum principle to investigate the nature of optimal controls for time, fuel and energy optimal problems. We also pointed out that the minimum principle yielded only a qualitative idea about the controls involved, while it did not explicitly yield the optimal controls as a function of the present state variables. The optimal controls obtained by using the minimum principle are generally a function of time for high order systems. It is easier to implement the optimal control laws if we know the control as a function of state variables, as the states may be readily measured. If the control as a function of state variables is not known, then it is still possible to implement the optimal control laws with control as a function of time, but such a system tends to be open loop and is unreliable. In the sequel, we would like to point out that generally for higher dimensional systems, a closed loop optimal control law is very difficult to obtain.

We have also noticed that the optimal control depends on the initial adjoint vector $\underline{\psi}^*(0)$, which in turn depends in some unknown manner on the initial state vector $\underline{X}(0)$. Unfortunately no formulae yet exist to relate $\underline{\psi}^*(0)$ to $\underline{X}(0)$. The procedures that are available are iterative in nature. It should be pointed out that realization of optimal controls by using an on-line computer to calculate $\underline{\psi}^*(0)$ for every $\underline{X}(0)$ and then generating the optimal control in real time may not be feasible in critical applications, such as the re-entry vehicle problem. Moreover, the processing computer system would be very complex. If the initial guess of $\underline{\psi}^*(0)$ is not wisely chosen then it may take considerable time for the computer to obtain the final value of $\underline{\psi}^*(0)$. Moreover, in order to calculate $\underline{\psi}^*(0)$ we must know the exact system dynamics. In practice the systems involved are non-linear and may also sometimes defy proper characterization.

2.2 Learning Machines and Artificial Intelligence:

In order to overcome the difficulties described above we can suggest several alternatives. For example, we can collect extensive data about the optimal controls for state vectors distributed uniformly in the state space of interest and store it in a memory device. Such a collection of data can be done by actual computation on a computer

or, by ground simulation if the system dynamics is not very well-known. Application of such a method would not be economical as it would require a large memory storage. Besides there may arise difficulties such as the state variables in question not belonging to the data set stored in the memory.

An alternative approach is to use an "Intelligent Controller". By "Intelligent" we mean one that has the capability to "learn" a solution to the optimal control problem, and is capable of giving a favourable response by associating the event in question with events that are familiar. These so-called "Intelligent Controllers" are expected to have properties similar to human learning.

As far as the latter is concerned we can state that based on human experience the skill to perform a job depends in general on the training and experience, and it is because of man's adaptive nature that he is able to learn. In this context one may also ask the question: "Why are persons with more "experience" preferred over others?" It is because experienced people can take decisions more quickly and wisely than those who have less experience. The experienced persons have been found to have creative abilities and ability to handle situations, unseen in past.

Again, if we consider some of the classic experiments carried out by Pavlov [7] on animals, it was noticed that the salivation which occurs in response to the view or smell of food, can also be elicited by a different stimulus such as by ringing a bell, if the ringing precedes or accompanies the presentation of food for a number of trials. He called these responses to the substitute stimuli, "Conditioned Reflexes", in the present use the term applied is conditioned response. Pavlov wrote in 1927 [7]:

"..... if the intake of food by the animal takes place simultaneously with the action of a neutral stimulus which has been hitherto in no way related to the food, the neutral stimulus readily acquires the property of eliciting the same reaction in the animal as would the sight of food itself. This was the case with the dog employed in our experiment with metronome ....

..... after several repetitions of the property of stimulating salivary secretion and of evoking the motor reactions characteristic of elementary reflex."

We note that the critical relationship in the experiment was the time relation between the original stimulus (unconditioned) and the neutral (conditioned) stimulus. The two had to occur close together in time with conditioned stimulus occurring ahead of unconditioned stimulus. We find

here that the dog had adapted itself to associate the ringing of bell with the service of food. We also note that such a behaviour was obtained with repetitive trials. Each time the trial took place, the unconditional stimulus was getting reinforced.

Learning has been defined by psychologists as: the process by which behaviour is originated or changed through practice or training.

Thus given a goal and a situation, an animal can be conditioned to some response. It has been observed that reinforcement by reward or punishment are important factors in hastening the learning process.

Similarly our objective is to construct machines which can change their internal structure in such a way that its response to a certain situation would be correct, had its previous response to the same situation been incorrect. It is desired that the response of the machine improves over successive trials.

The function of an intelligent controller is:

(i) to interpret the present event (the present state variables) by generating a response (control signal) by using its past experience.

(ii) if the present response is correct then either reinforce the attributes to the response, or just do nothing.

(iii) if the present response is incorrect then change the internal structure in such a way that on the next occurrence of the same event the response is correct.

An important question can now be asked, "Do such intelligent machines exist?" and if yes, "How can we construct them?" We may very well start with answering the question, "What is intelligence?" The dictionary [8] definitions are

(i) capacity for understanding; and other forms of adaptive behaviour and

(ii) knowledge of an event; circumstance etc.; received or imparted; news; information.

Discussions of the definition of intelligence and artificial intelligence have been in progress ever since the ability of the digital computer to perform logical operations was compared with the living brain.

If we consider intelligence to be a uniquely human attribute then by definition, this attribute can not be associated with a computer or any other artificial system.

On the other hand if one argues that there is some fundamental physical limitation on the ability to compute, process, decide, and think which applies to the creation of machine intelligence, then these same limitations must also apply to human intelligence. That is, any theoretical limitations based on universal natural laws must be equally applicable to man and machine.

A more scientific approach to this question was proposed by Turing [9] in 1950. Turing suggested that the question whether the digital machines are as intelligent as the human brain can be answered by experiment and observation in which the relative behaviour of a system (which by definition, thinks and exhibits intelligent behaviour - namely human behaviour)is observed. Turing proposed a question-answer scheme in which a person could ask questions and receive answers via some remote terminal such as a teletype and the questions could be answered either by a digital machine or another man at the other terminal. If one could determine whether a man or a machine is answering the questions then one may question whether machines are as intelligent as man. On the other hand if one could not determine whether a man or a machine is answering the questions, then one may say that machines are as intelligent as man. This imitation game is one approach which has not yet been fully explored.

Another common argument against machines being intelligent is that a computer can do only what it is told to do by its human programmer. But if we really examine the nature of human intelligence, then we are forced to admit that due to the adaptive nature of human brain or the capability to learn, the inherent tendency to survive, and his experience about nature, give him the powers of thought

and intelligence. The basic, built in characteristics of man coupled with the changes in him through experience generate the proper "program" for successful performance in his environment, this "program" being subject to changes. The present intelligence in man may be through some evolution process, obtained by the experience of ages.

Simon [10] has suggested: "If a computer thinks, learns, and creates it will be by the virtue of a program that endows it with these capacities. Clearly this will not be a program – any more than the human is – that calls for highly stereotyped and repetitive behaviour independent of the stimuli coming from the environment and the task to be completed. It will be a program that makes the system's behaviour highly conditioned on the task environment – on the task goals and on the clues extracted from the environment that indicates whether progress is being made toward these goals. It will be a program that analyzes, by some means, itsown performance, diagnoses its failures, and makes changes that enhance its future effectiveness".

2.3 Foundations of Learning Machines:

Shannon [12] had demonstrated that abstract logical operations such as addition, multiplication, negation, implication and equivalence could be implemented

physically by using electrical switching circuits. The basis of the suggested implementation is the fact that to each combination of abstract logical propositions linked by connectiveness like "Not", "And", or "Or"; there corresponds a circuit formed by connecting basic components such as the inversion (or negation), addition and multiplication switches. Shannon's work was the foundation of the modern digital computer theory.

Later during 1943, McCulloch and Pitts [13], who were studying neurons* and neural networks, became interested in using threshold logic elements like biological neurons in place of ordinary switches in logical networks. They were interested in finding out whether threshold logic elements can be used to realize various switching functions. Above all, they believed that since the living brain also works on the principles of logic (all-or-none principle), hence it should be possible to explain the intelligent behaviour of the brain by studying simpler networks with neurons. They found out that such networks with threshold logic elements could be used with ease in realizing the logical operations. A very remarkable fact was that with neural networks having

---

* Neurons are nerve cells which are functional units of brain. The cell body has several inputs called "dendrites" and the output is a thin long fibre called "axon". A detailed discussion of neurons can be found in Eccles [11].

adjustable or alterable synapse* (so as to lower or raise the threshold value), the same network could realize an entirely new logical switching function with a new threshold value. This was the foundation of a new kind of machine which exhibited adaptive behaviour. The networks of neurons with adjustable synapse acquires a means of changing the mode of their internal organization to yield the prescribed behaviour even under widely varying conditions. As a result, these can be made to mimic Homeostasis (self adjustment in changing environment, [14]). Hebb [15] has suggested the following idea to explain adaptation and learning in the living brain, "when an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased". Consequently, permanent changes in the threshold excitation of the neuron can possibly occur, although neurophysiologists have not yet been able to demonstrate them in the brain.

Thus we find that the idea of adjustable synapse proves to be useful one. Using this principle we can construct

---

* Synapse is the point where an axon from a neuron ends on a dendrite of some other neuron.

networks of neuron-like structures to achieve adaptive behaviour. In the next section we shall describe some basic concepts of pattern recognition which are necessary for understanding the principles of learning control systems.

2.4 The Pattern Recognition Problem: Introduction:

Mathematically, pattern recognition is a classification problem. Consider for example the recognition of characters. We wish to design a system such that a hand-written symbol will be recognized as an "A", a "B" etc. In other words we wish to design a machine that will classify the observed character into one of 26 classes. The hand-written characters are often ambiguous, and there will be misclassifications of the characters. The major goal in designing a pattern recognition machine is to have a low probability of error.

There are many problems that can be formulated as pattern recognition problems. For example, in weather prediction, the weather may be divided into three classes; fair, rain and possible rain, and one may like to predict the weather at a future date into one of these three classes. In the recognition of electrocardiograms, the classes are disease categories and the class of normal subjects.

The input to a pattern recognition machine is a set of measurements, and output is the classification. For example in weather prediction, possible measurements could be atmospheric pressure humidity, temperature etc.

In pattern recognition terminology these measurement variables are called "Features". Generally one might expect to include all the possible features as input variables for a pattern recognition machine, but it may be "costly" to include all features. It has been shown by Hughes [16] that as the number of features is increased beyond certain limit, the performance of the pattern recognition machine deteriorates. Moreover, some of the features may not be relevant at all, or some features may be dependent on some other features. For example, in the case of weather prediction the proposed features could be temperature, pressure, month of the year, humidity, the wind velocity or the condition of sky. But the relevant features could be temperature, pressure, month of the year or wind velocity, as we know humidity depends on temperature and thus it is irrelevant.

Apart from feature selection it may be necessary to process the input variables, to provide a convenient input format. By processing, the problem of pattern recognition becomes easy.

In pattern recognition, we assume that each set of data to be classified is a set of n real numbers, s_1, s_2, ... s_n. We call such a set a pattern. A device which sorts patterns into categories is called a pattern classifier.

A general block diagram of a pattern recognition system is shown in Fig. 2.1.

To illustrate further some more ideas of pattern recognition consider a lumber mill producing assorted hardwood planks that wants to automat the process of sorting finished lumber according to the kind of tree. If it is required to sort teakwood from rosewood, we may check whether the wood sample is hard or soft, or whether it is coarse grain or fine grain or even by checking its weight. All these properties are the features of a wood sample. One method is shown in Fig. 2.2.

If we take a number of pieces of rose and teak woods, and consider their weights and the colour of wood as features, then we might notice that rosewood is often darker than teakwood and also heavier. Thus we may have a rule for deciding the sample as rosewood if its weight is more than a certain critical value or its colour is darker than a critical value. If weight of wood can be measured by an index s_1, and the darkness by some other index s_2, then we call the vector $\underline{S} = [s_1, s_2]^T$ as input feature

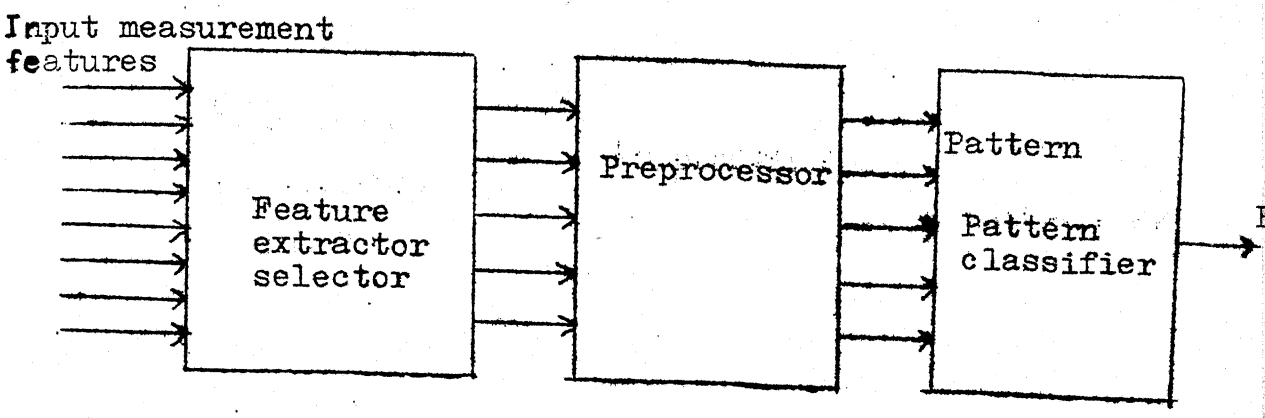


Fig. 2.1 A general pattern recognition system

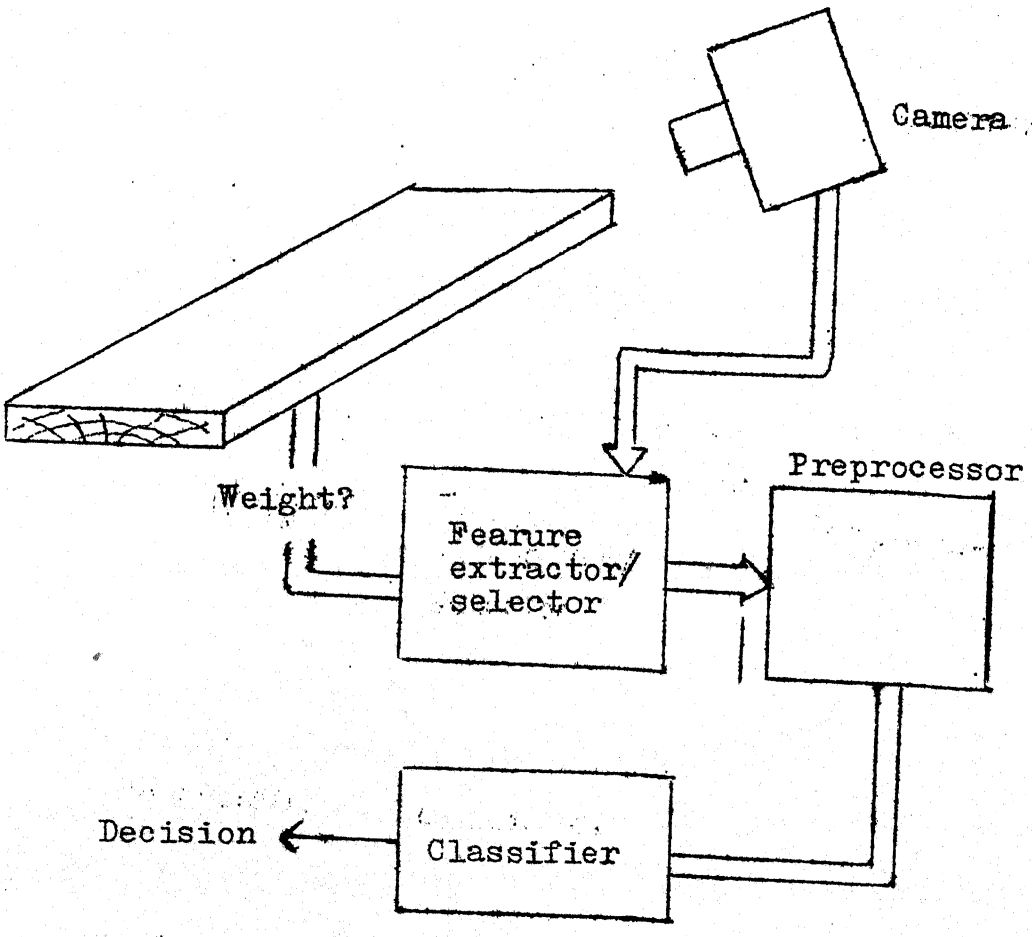


Fig. 2.2 A wood classification scheme

vector. Taking several wood samples and measuring the feature vectors and plotting them on a graph, the representation might appear as in Fig. 2.3.

Notice that the points corresponding to rosewood cluster distinctly as against the cluster of points corresponding to teakwood. Thus we can form a rule of decision that if a feature vector occupies a position above a line separating the two clusters then we can decide the sample as rosewood, and on the contrary as teakwood. In other words the problem is to partition the feature space into two regions, where all the points in one region correspond to rosewood and the other region corresponds to teakwood. In the above case we considered only two features. We may be faced with the situation that a sample of rosewood exhibits the same features as that of a sample of teakwood. In this case we might like to introduce another ground of classification, for example, the straightness of grain. Thus in order to increase the accuracy of classification we would like to include several features. In such a case the partitioning could be achieved by means of a hyperplane. When the set of points can be partitioned by means of a hyperplane, then we call these points as linearly separable, and the classifying machine is called a linear machine. If, in order to partition the pattern space more than one hyperplane becomes necessary,

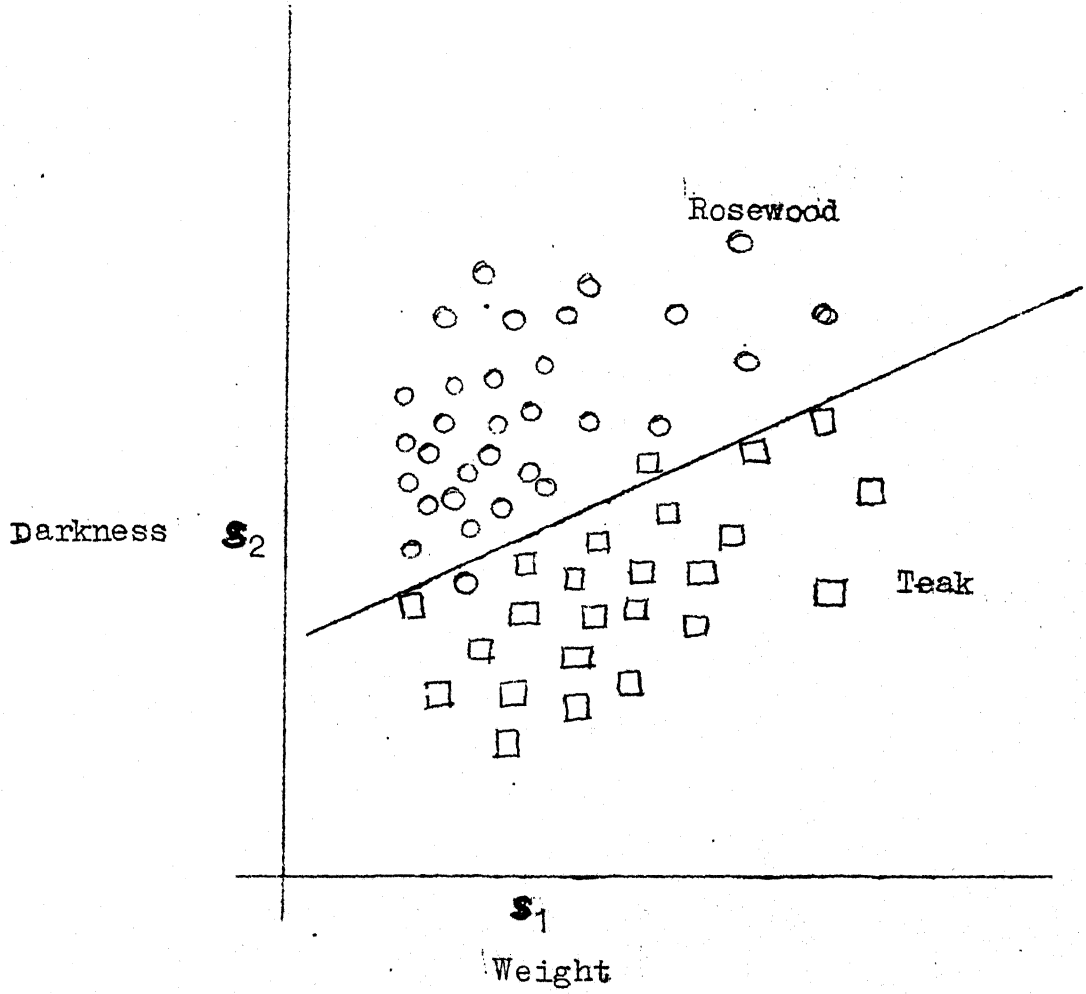


Fig. 2.3 Scatter-diagram for feature vectors in a wood-classification scheme

then the samples are piecewise linearly (or nonlinearly) separable. The classifier is called a piecewise linear machine. The rule of decision and the mathematical expression describing it in feature space is called a Discriminant function.

2.5 Some Basic Ideas in Design of Learning Control Systems:

In the previous section we examined the foundations of self-organizing and learning systems. In the present section we shall examine how these ideas can be used in designing learning control systems.

A self-organizing system contains the following four basic subsystems, as depicted in Fig. 2.4: (i) Sensors, (ii) Preprocessor (iii) Learning network (iv) Goal circuit.

The sensors observe the environment and provide the descriptive data for the learning network and goal circuit. The preprocessor processes the data from the sensors so as to match the requirements of learning system, e.g., the input state variables may be quantized and then these quantum states may be coded in a convenient form. In this way the complexity of the problem is reduced as only a finite number of quantum states have to be considered. The learning network is a variable part of the self-organizing system. It may be a device which changes its function chemically or

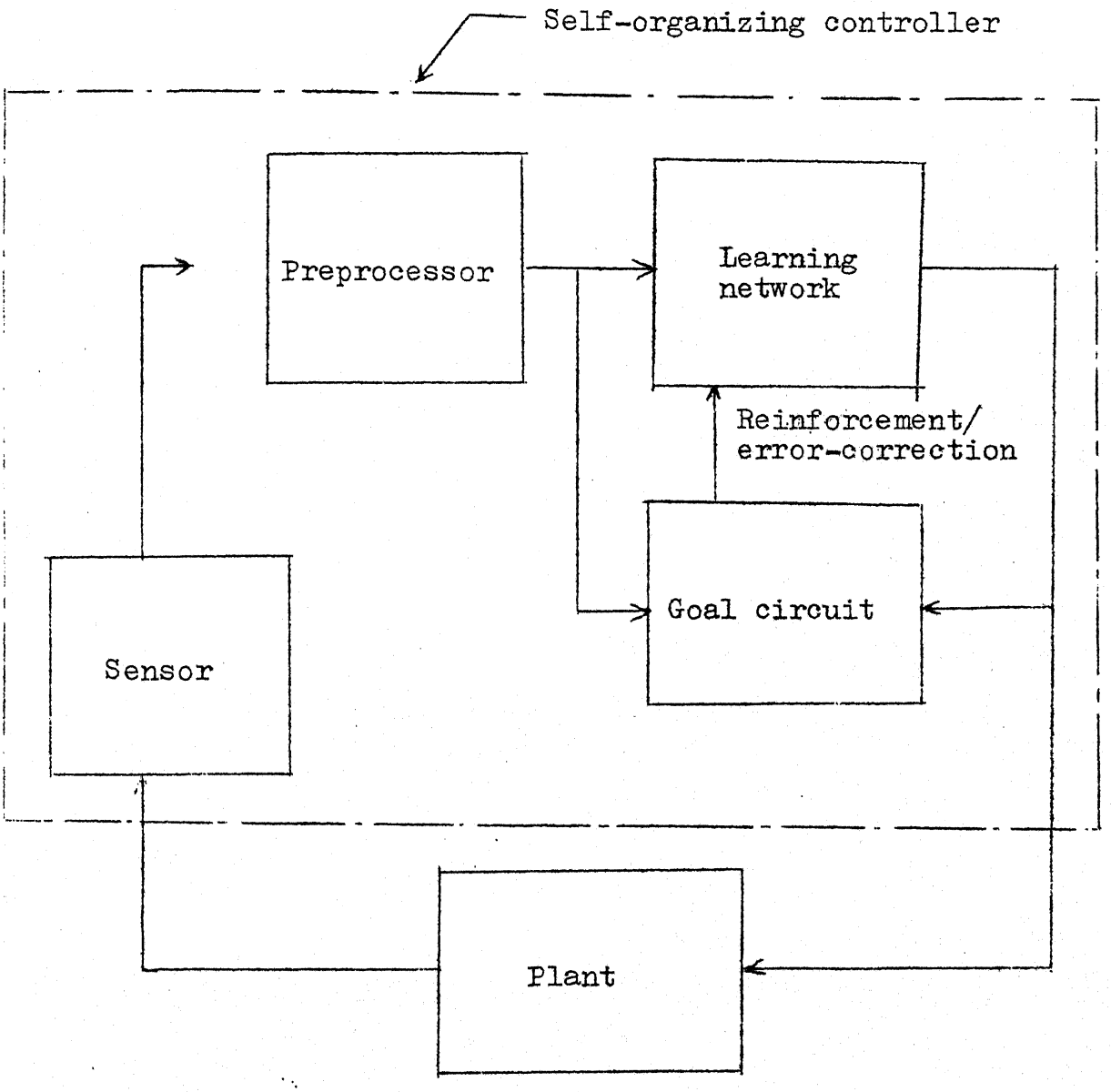


Fig. 2.4 A self-organizing control system

physically, or it may just be a part of a computer algorithm. At present only the first and the last type of learning networks are comprised of decision elements which operate on processed data and render a desirable response. The goal circuit directs the system organization toward a specific objective and provides information on the degree of success attained at the end of each trial. This is usually achieved by one of the following techniques.

(i) Reinforcement: If the present performance is an improvement over recent past performances, then it generates a reward signal to reinforce those states of the learning network that constitute the improvement and on the other hand weakens those states due to which the present performance gets deteriorated by generating a punishment signal.

(ii) Error-correction: A correcting signal is generated by the goal circuit to weaken those states of the learning network only when the present performance is worse than recent past performances.

At the heart of learning systems there are four basic concepts, referred to as artificial intelligence techniques. These are, (a) Mapping (b) Control situations (c) Memory (d) Subgoal. We shall briefly describe these.

Mapping: The mapping concept permits one to view control system design problems as collections of mappings from the state space to the control category space. Such a concept allows us to formulate the design problem in a form which is suitable for pattern recognition interpretation.

Control situations: Control situations are regions in the state space for which a single control choice (e.g. a control +1 or -1) leads to satisfactory performance for all points contained therein. Such regions are obtained by pregridding the state space (e.g., quantizing the state variables). The basic idea is that neighbouring points within a small region should have the same control choice with high probability.

Memory: With the memory concept a separate memory location is associated with each control situation. As learning proceeds, it is necessary to store the pertinent information about the control situations encountered in the past. Thus if a particular control situation encountered in past occurs again the learning continues from the learned state attained in the past. In other words with the aid of memory during the learning process the learning system takes advantage of the past "experiences", and thus tends to improve its future performance.

Subgoal: Often the main goal or objective for a control system can not be formulated precisely, or if it can be, the formulation is mathematically intractable. A common approach is to formulate a subgoal that approximates the actual goal.

2.6 The ADALINE and Trainable Controllers:

The Trainable Controller learning control system is one in which the inputs and plant are incompletely specified. The actual controller (Teacher) may be a set of control situations with a known control choice for each. The learning system "observes" the activities of the "Teacher" and tries to copy them, i.e. for each control situation it tries to generate a control using its past "teachings" and compares them with that of the teacher. If the performance is bad then it changes its organization in order to improve its performance. In Fig. 2.5 a Teacher-Learner Trainable control system is shown.

The "Teacher" controls the plant by generating optimal controls for a small set of initial conditions $\underline{X}(0)$ of the plant. While the "Teacher" controls the plant, learning system learns the optimal control law. Learning occurs with the switches S1 and S2 in the position 1 and 1' respectively. After learning is complete switches S1 and S2 assume the position 2 and 2' respectively. In this mode the learning system controls the plant.

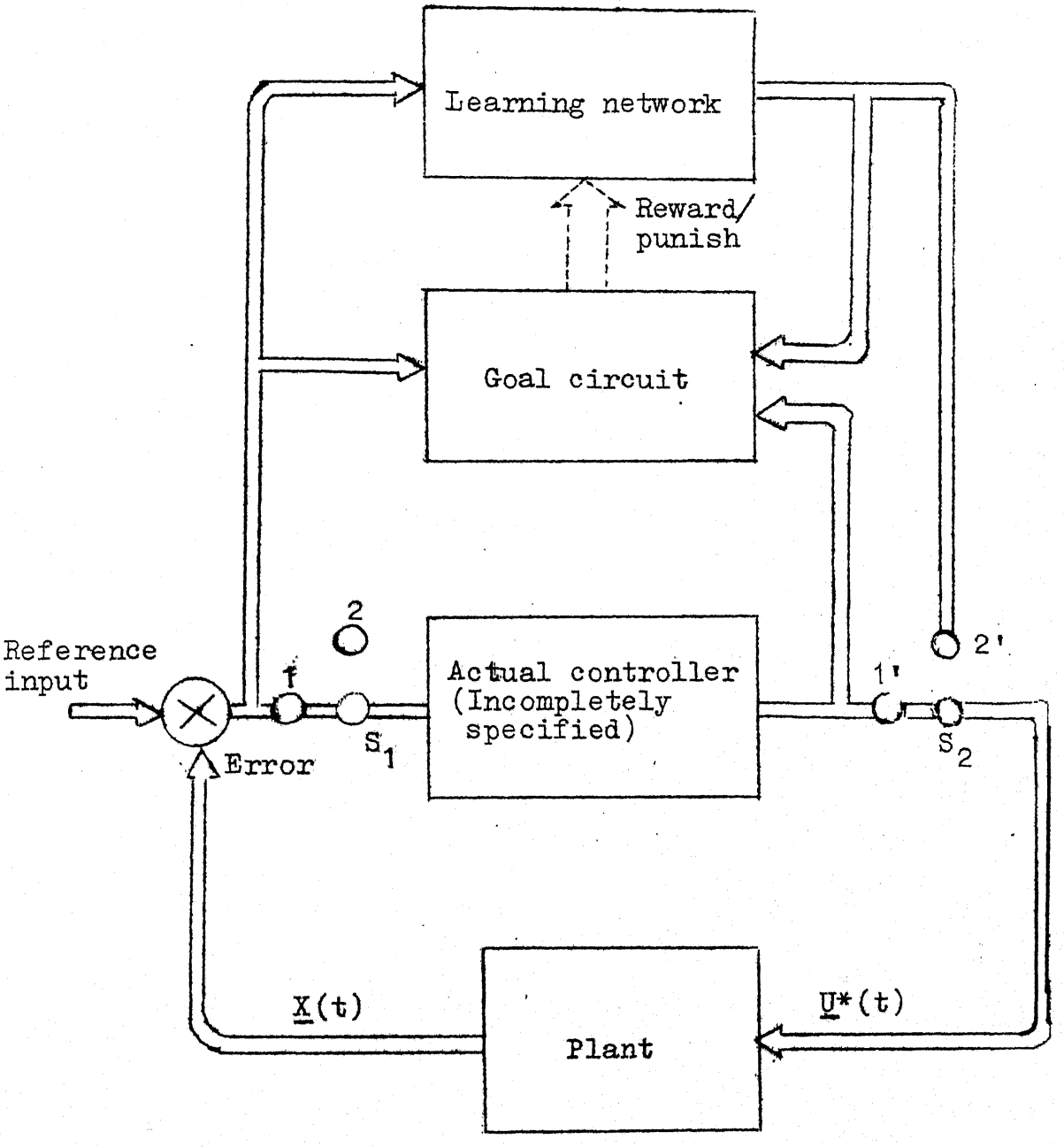


Fig. 2.5 A teacher-learner trainable control system

The basic building block of a learning system is an ADALINE (ADAptive LInear NEuron). This is a simplified hardware copy of a biological neuron. Its structure was proposed by Widrow [17] in 1962. A representative diagram of an Adaline is shown in Fig. 2.6.

The Adaline consist of a relay unit R, a summing device S, variable weights w_0, w_1 ... w_n and an adaptation machinery. The numbers s_1 ... s_n are real numbers and a set of such numbers is called a pattern or stimulus. It is assumed that the elements of a stimulus appear at the specified terminals at the same instant. The pattern elements may take values (0,1) or (-1,1). The adaptation machinery can vary any weight independent of the other weights.

We describe a threshold device by,

$$\left.\begin{aligned} &\sum_{i=1}^{n} s_i . w_i > T \text{ , then output response } R = +1 \\ &\sum_{i=1}^{n} s_i . w_i < T \text{ , then output response } R = -1 \end{aligned}\right\} \quad (2.1)$$

where T is a number called the threshold, s_i are elements of the stimulus and w_i are the weights.

Thus in order to get an output +1 the weighted sum of the inputs should exceed a threshold value. Note in Fig. 2.8 that there is a weight w_o. By adjusting this weight the proper threshold value is obtained. Note too that

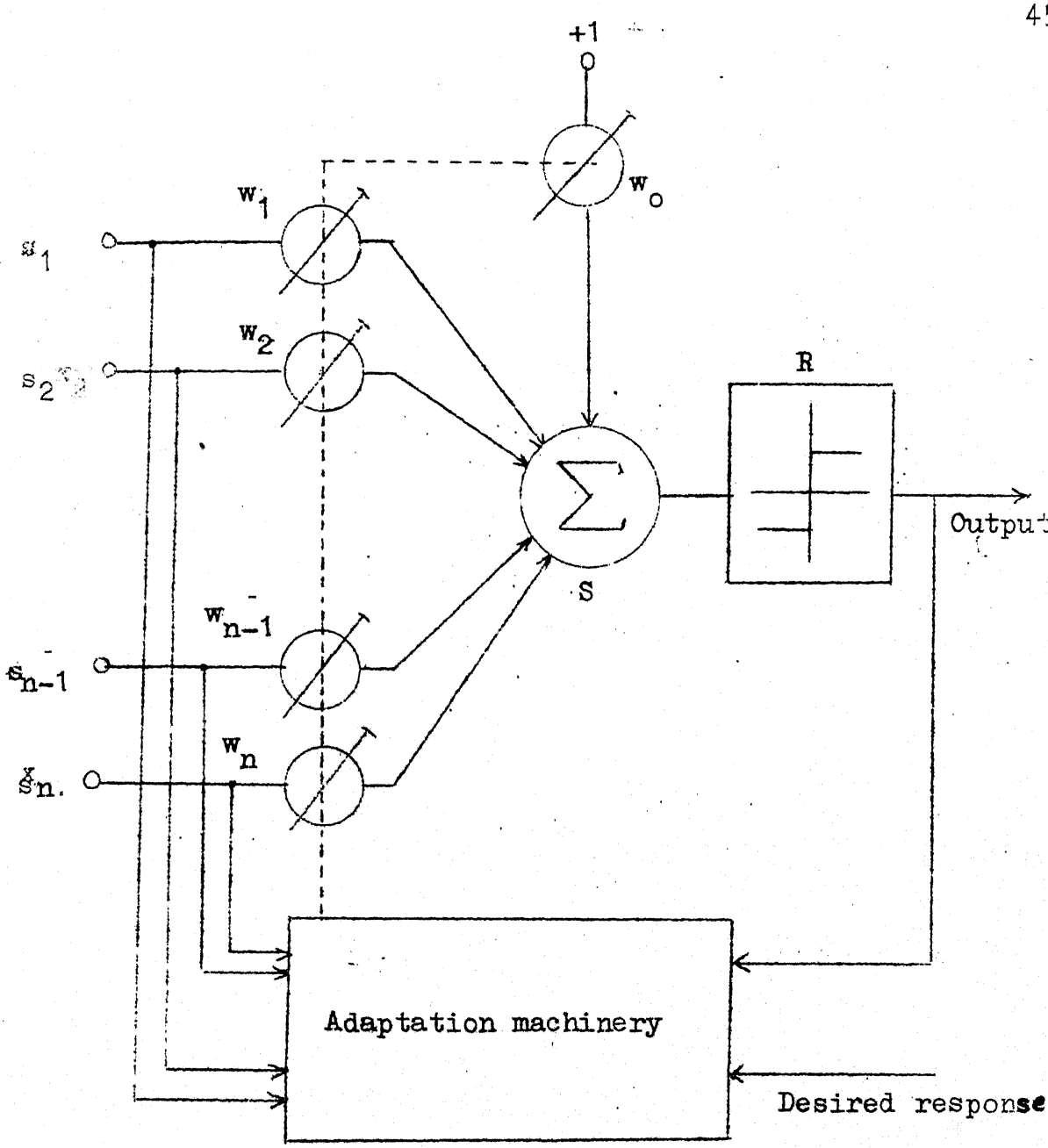


Fig. 2.6 A basic Adaline.

the input to this weight is +1. If we replace T by w_o in (2.1) we obtain

$$\left.\begin{aligned} &\sum_{i=1}^{n} s_i . w_i \geq w_o . 1 \ , \quad \text{then output} = +1 \\ &\sum_{i=1}^{n} s_i . w_i < w_o . 1 \ , \quad \text{then output} = -1 \end{aligned}\right\} \qquad (2.2)$$

In vector notation we have

$$\left.\begin{aligned} &\underline{W}^T \underline{S} > w_o \ , \quad \text{then output} = +1 \\ &\underline{W}^T \underline{S} < w_o \ , \quad \text{then output} = -1 \end{aligned}\right\} \qquad (2.3)$$

The vector equation,

$$\underline{W}^T \underline{S} - w_o = 0 \qquad (2.4)$$

has a special significance. since it represents a hyperplane in the pattern space. A pattern vector $\underline{S}$ satisfying (2.4) lies on the hyperplane. For the patterns not lying on this hyperplane, value of the left hand side of (2.4) is either positive or negative, depending on which side of the hyperplane the pattern lies. In other words the hyperplane represented by (2.3) partitions the pattern space into two regions.

Modifing (2.4) as

$$\sum_{i=0}^{n} s_i w_i = 0 \qquad (2.5)$$

or in vector form,

$$\underline{W}^T\underline{S} = 0 \qquad (2.6)$$

where $\underline{W}$ and $\underline{S}$ are such that

$$\underline{W} = \begin{bmatrix} w_o & w_1 & \cdots & w_n \end{bmatrix}^T$$

and

$$\underline{S} = \begin{bmatrix} 1 & s_1 & s_2 & \cdots & s_n \end{bmatrix}^T$$

For fixed weight settings there are 2^n different combinations of the input pattern elements with binary values. Each of these patterns would cause a +1 or -1 Adaline input. Thus all possible input patterns are classified into two categories. The input-output relationship is determined by the choice of weights. We can imagine each pattern input being represented as a vertex of a unit-hypercube of (n+1) dimension. Thus adjustments of the weights of Adaline in fact correspond to orienting an n dimensional hyperplane passing through the origin. Only a subset of 2^n input-output relationships are however realizable by an Adaline. The set of patterns which can be classified correctly by an Adaline are called linearly separable patterns.

Considering (2.6) we have the following rule of decision. If

$$\begin{aligned} &\underline{W}^T\underline{S} > 0 \text{ , then the class of pattern } = +1 \quad \text{(a)} \\ &\underline{W}^T\underline{S} < 0 \text{ , then the class of pattern } = -1 \quad \text{(b)} \end{aligned} \qquad (2.7)$$

Forming a matrix A by combining the rows of all input patterns which are linearly separable, then for a particular partition we have

$$A = \begin{bmatrix} P^+ \\ \hline P^- \end{bmatrix} = \left[\begin{array}{cccc} +1 & s_{11}^+ & s_{1n}^+ & \cdots \; s_{1n}^+ \\ +1 & s_{21}^+ & s_{22} & \cdots \; s_{2n}^+ \\ \cdots & \cdots & \cdots & \cdots \\ +1 & s_{l1}^+ & s_{l2}^+ & \cdots \; s_{ln}^+ \\ \hline +1 & s_{l+1,1}^- & s_{l+1,2}^- & \cdots \; s_{l+1,n}^- \\ \cdots & \cdots & \cdots & \cdots \\ +1 & s_{l+m,1}^- & \cdots & s_{l+m,n}^- \end{array}\right] \qquad (2.8)$$

where P^+ and P^- are associated with the two classes of vertices defined by (2.7-a) and (2.7-b) respectively. The number of patterns belonging to + class and - class are l and m respectively. Hence,

$$l + m \leq 2^n \qquad (2.9)$$

We notice that

$$P^+ \underline{W} > \underline{0}$$

and $$P^- \underline{W} < \underline{0},$$

where $$\underline{0} = [0 \;\; 0 \;\; \ldots \;\; 0]^T \qquad (2.1$$

If we multiply all the entries corresponding to P^- in the matrix A by -1, then we have

$$\bar{A} = \left[\frac{P^+}{-P^-}\right] = \left[\begin{array}{cccc} +1 & s^+_{11} & \cdots & s^+_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ +1 & s^+_{l,1} & \cdots & s^+_{l,n} \\ \hline -1 & -s^-_{l+1,1} & \cdots & -s^-_{l+1,n} \\ \cdots & \cdots & \cdots & \cdots \\ -1 & -s^-_{l+m,1} & \cdots & -s^-_{l+m,n} \end{array}\right] \qquad (2.11)$$

and

$$\bar{A}.\underline{W} > \underline{0} \qquad (2.12)$$

Let us assume that we know a closed form solution for optimal control in a time optimal control system, i.e. we know a switching surface in the state space described by the equation

$$F(x_1, x_2, \ldots x_k) = 0 \qquad (2.13)$$

where x_1, x_2 ... x_k are the state variables.

Now if (2.13), does not contain any cross product terms then we have,

$$F(x_1, x_2 \cdots x_k) = \sum_{i=1}^{k} f_i(x_i) = 0 \qquad (2.14)$$

If we quantize each state in q finite quantum levels, then we quantize the state space under consideration in q^k quantum control situations. If we assign a code to each quantum level of each state then (2.14) becomes

$$F(\underline{X}_1, \underline{X}_2, \cdots \underline{X}_k) = \sum_{i=1}^{k} f_i(\underline{X}_i) = 0 \qquad (2.15)$$

where $\underline{X}_i$ is a vector representing a coded quantum level.

Strictly speaking (2.15) approximates the original switching surface due to quantization errors. We can ignore this error since it can be made arbitrarily small.

Fig. 2.7 shows a modified form of an Adaline along with a quantizer-encoder for each state variable. It should be noticed that it does not make any difference whether the weighted sum of all the patterns is taken or the weighted sum of individual state patterns is taken first and then the final sum is taken. The new individual threshold weights are such that the sum of these equals the original threshold weight. If the linear element has to mimic the actual time optimal surface, then in each quantum control situation at least at one point should be considered of the form

$$f_i(x_i) = \underline{X}_i^T \underline{w}_i \qquad (2.16)$$

Now let Q be the code matrix the rows of which are the codes assigned to the quantum levels of any state.

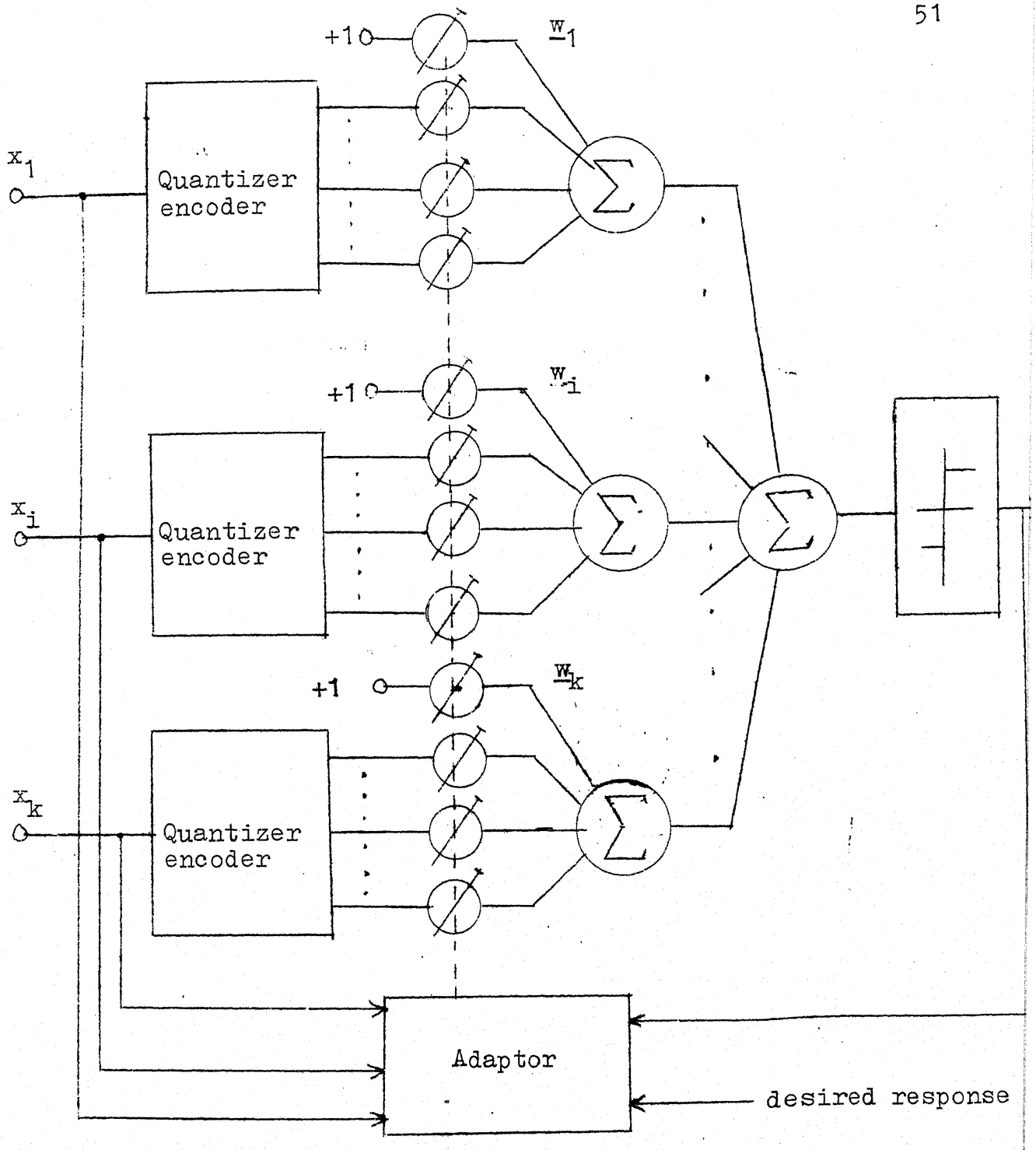


Fig. 2.7 A modified Adaline.

Thus if the Adaline mimics for all quantum levels of the state variable x_i, then

$$Q\,\underline{w}_i = \underline{f}_i(x_i) \tag{2.17}$$

where $\underline{f}(\underline{x}_i) = \left[f_1(\underline{x}_i) \ \ldots\ f_q(\underline{x}_i)\right]^T$

Now in (2.17) if Q has rows which are linearly independent, then Q^{-1} exists, hence we have

$$\underline{w}_i = Q^{-1}\,\underline{f}\,(x_i) \tag{2.18}$$

In other words if the codes are linearly independent, then a solution for $\underline{w}_i$ exists provided no cross terms exist in (2.13).

The other criteria applicable to the realization of switching surfaces have been developed by Berkoveck and Epley [18]. These criteria are in terms of a property called projectability, which is defined: A surface that is continuous and single valued along any line parallel to a direction in space is projectable along that direction.

Two tests for this property are:

(1) A surface in n-dimensional space with a continuous boundary, symmetrical about an (n-1) dimensional hyperplane is projectable along a direction perpendicular to the hyperplane if the boundary is single valued on one side along any line side along any line normal to the hyperplane of symmetry.

(2) A surface in n-dimensional space with a monotonic boundary, i.e. for which

$$\frac{\partial F}{\partial x_i} \leq 0 \quad \text{or} \quad \frac{\partial F}{\partial x_i} \geq 0 \text{ , for any } i = 1, \ldots k$$

is projectable.

The theorems proposed by Berkoveck and Epley are:

(a) A surface which is projectable may be approximated with arbitrary accuracy with a single Adaline by using a linearly independent code for each of the state variables.

(b) Any region which may be transformed into a projectable region with a linear transformation is realizable with a single Adaline.

These are sufficient conditions but not necessary.

Finally we may point out that the criteria stated above needs the analytical expression or the geometry of the switching surface. If, however, these are known, off-line-training would be of little use.

2.7 Storage Capacity of Adaline:

From the last section we observe that there are 2^n possible patterns for n-dimensional patterns. There are 2^{2^n} possible partitions of the 2^n patterns in different ways. Several attempts have been made to find out a relation to calculate the number of linearly separable partitions out

of 2^{2^n} maximum. Attempts have been successful upto n = 8 (Winder [19]). An upper bound has been estimated on the number of linearly separable partitions as,

$$U(n) = \frac{\sqrt{2}}{\pi n} \left(\frac{e2^n}{n}\right)^n \qquad (2.19)$$

where n = dimensionality of patterns.

In case of an incomplete training sample being specified (i.e., $1 + m \leq 2^n$ in (2.9), the curve in Fig. 2.8 indicates the probability of realizing a randomly selected partitioning hyperplane. The abscissa is normalized and represents the number of patterns in the sample per adaptive weight i.e. $(1 + m)/(n + 1)$. The ordinate Ps is the a-priori probability of achieving a randomly selected partitioning hyperplane of n-dimension, when using an Adaline with (n+1) variable weights.

We notice that for large n a threshold for Ps occurs at $(1 + m)/(n + 1) = 2.0$. This point is defined as the normalized capacity of an Adaline, and it means that for large values of n, an Adaline is almost certain to learn the classification of patterns in number less than 2(n+1) and almost certain not to learn patterns more than 2(n+1); patterns being selected randomly.

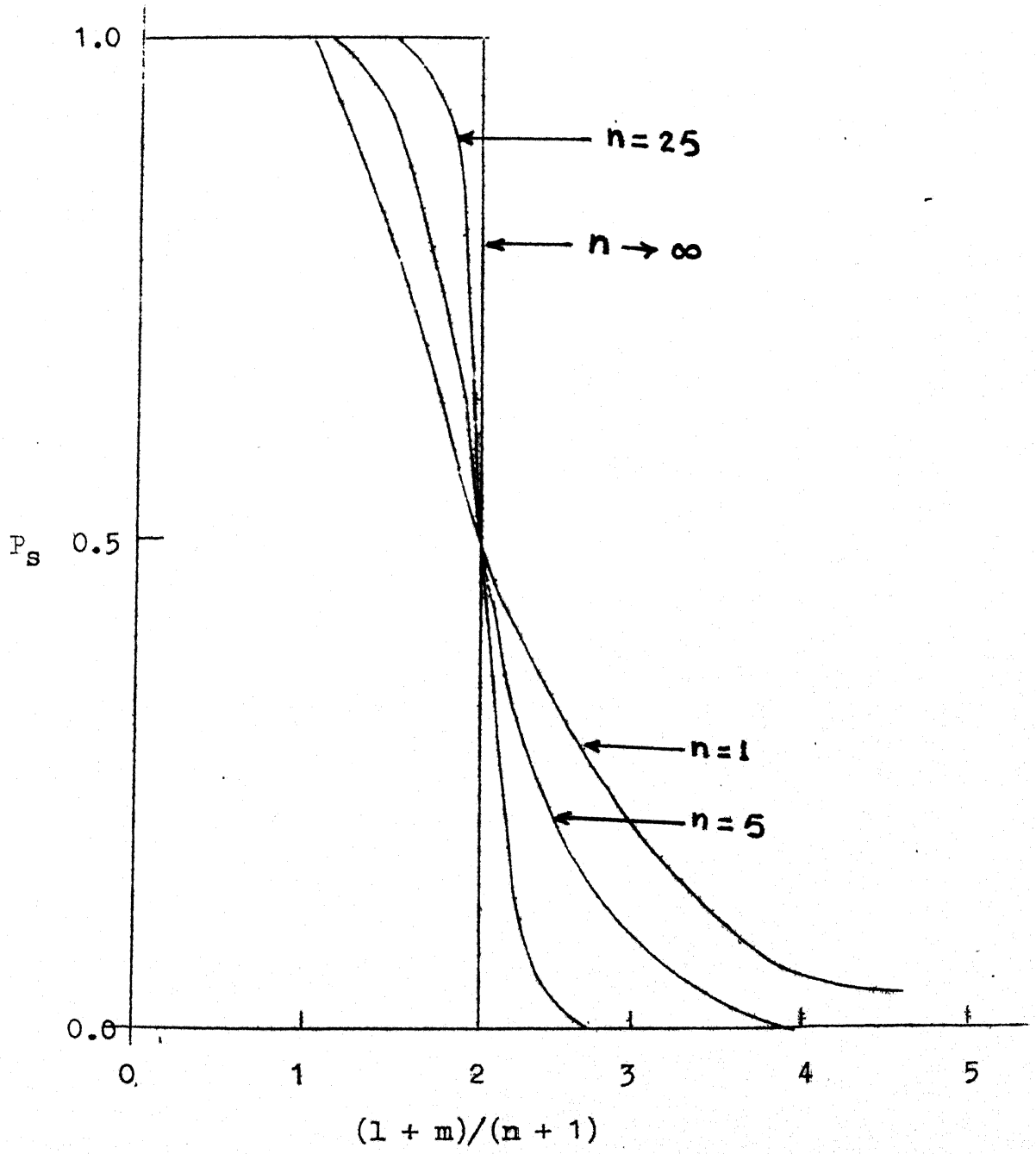


Fig. 2.8 Normalized storage capacity of an Adaline.

2.8 The Trainable Controller:

As noted earlier the response of an Adaline is very similar to a time-optimal controller (i.e. bang-bang type). Accordingly it is readily used as a trainable controller for time-optimal control problems. A representative block diagram is shown in Fig. 2.5.

The trainable controller consists of an Adaline and a quantizer-encoder unit for each state variable as shown in Fig. 2.7. The teaching controller controls the dynamic system and supplies the 'desired' control input u* to the dynamic system and the Adaline during the training mode, with switches S_1 and S_2 as shown in Fig. 2.5. The Adaline learns the time-optimal control law and once the learning is complete the trained controller can be put to use with switches S_1 and S_2 in position 2 and 2' respectively.

2.9.1 Quantization-encoding of state variables:

For the purpose of illustration let us consider an arbitrary switching surface with two state variables, as shown in Fig. 2.9. The switching curve shown in Fig. 2.9 could arise in a typical time optimal control problem.

After the state variables X and Y are quantized into a finite number of quantum levels as shown in Fig. 2.9, the state space under consideration is divided into several

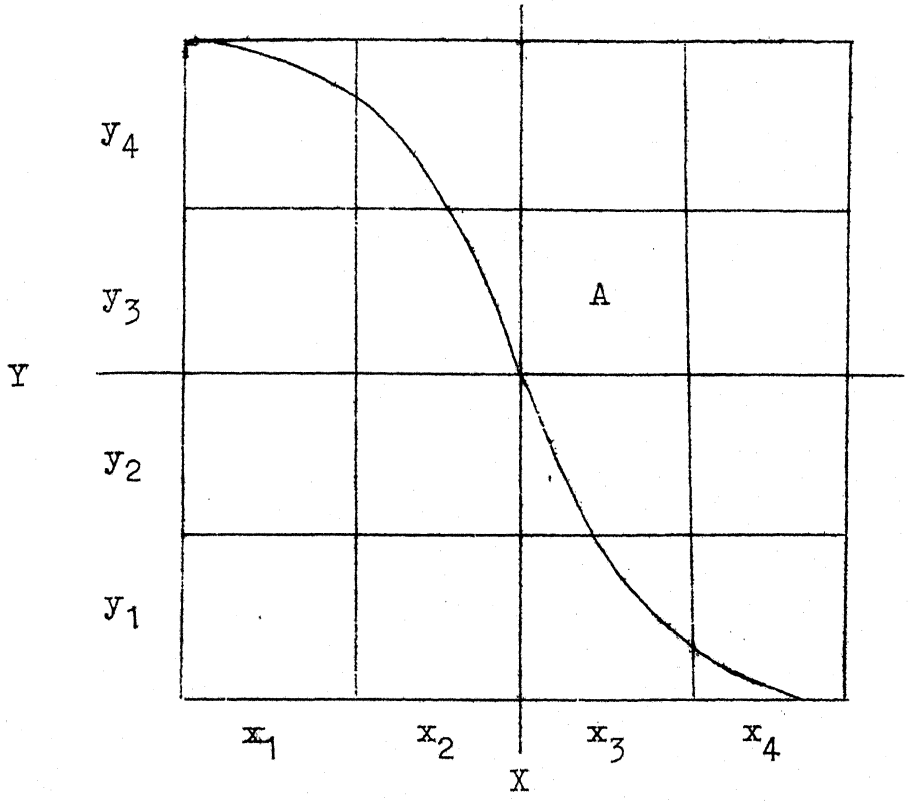


Fig. 2.9 A time optimal switching dichotomy in quantized state space.

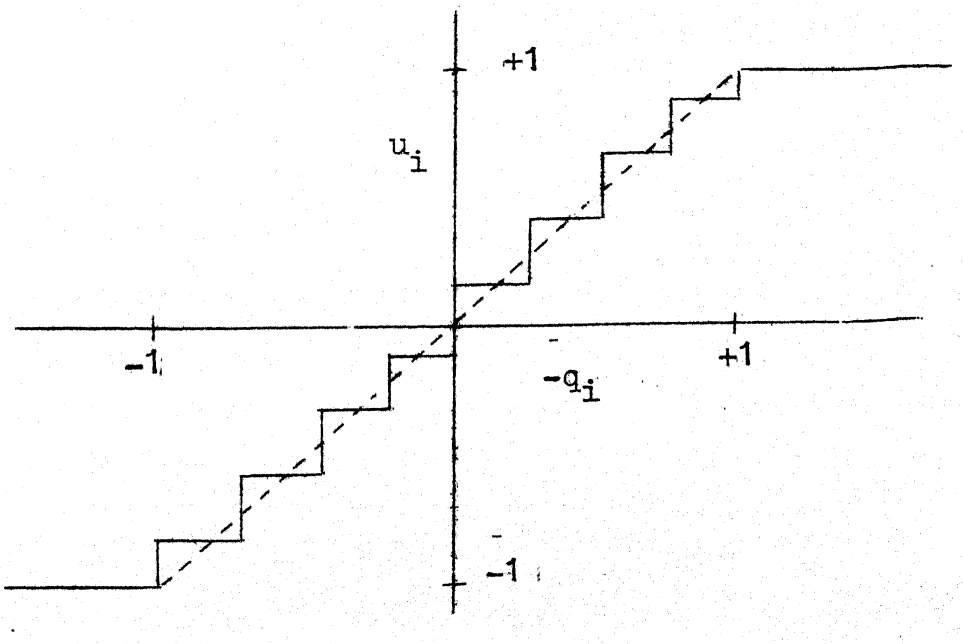


Fig. 2.10 Quantized energy optimal control.

To use an Adaline as a trainable controller for fuel and energy optimal control problems, a network of Adalines would be needed in general, since fuel and energy optimal controllers are multilevel in nature.

The quantization-encoding of state variables for fuel and energy optimal control problems is similar as discussed above.

For a fuel-optimal controller we need a minimum of two Adalines whose responses form a two bit word which can be decoded to identify the correct level i.e. +1, 0, -1. In case of energy optimal controller, the optimal-control can be quantized into n quantum levels as shown in Fig. 2.10. Thus in case of energy optimal controller a minimum of (n-1) Adalines is needed for n quantum levels of control signal. We shall describe in detail the use of Adalines for fuel and energy optimal control problems in Chapter 3.

CHAPTER 3

APPLICATIONS OF LEARNING SYSTEMS TO OPTIMAL CONTROL PROBLEMS

3.1 Introduction to Earlier Work:

In the previous chapter we discussed the method of learning control systems in obtaining solutions to the problems posed in Chapter 1. We noticed that when an optima control problem is viewed as a pattern recognition problem, the concepts of artificial intelligence can be usefully employed in synthesizing optimal controllers. We will notic that by the use of such techniques a controller so designed would be a quasi-optimal controller, since the switching surface it would realize, would be an approximation to the actual optimal switching surface due to quantization errors. The performance of such a controller would be somewhat suboptimal but, we find that its implementation would be more feasible.

Earliest work in this area is due to Smith [20] who studied the use of trainable controllers with threshold logic networks for third and fourth order, time-optimal systems. In this study the state space was quantized into hypercubes and those with same input control description were collected by using a network of threshold logic

elements. The designed controller was not tested for generalization as all the hypercubes were treated as training samples.

Subsequently Smith [21] has also considered the conditions for the realizability of switching surfaces. The analysis was carried out for the quantization-encoding of feature variables. It was shown that the sufficient condition for an Adaline to realize a given linearly separable dichotomy is that the quantum hypercubes be encoded with "linearly independent code".

Mendel [22] has made a detailed study of feasibility of using trainable controllers in optimal control systems. In his study, the system under consideration was a simplified model of a time optimal third order re-entry vehicle. The system dynamics being as follows.

$$\dot{\underline{X}}(t) = A\,\underline{X}(t) + \underline{b}\,u(t)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and

$$\underline{b} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T \qquad (3.1)$$

$u(t)$ being a scalar control input with the constraint

$$|u(t)| \leq 1$$

The problem was to design a controller which would drive the system from an initial state $\underline{X}(0)$ to a final state $\underline{X}(t_f) = \underline{0}$ in minimum time.

The state space of interest was quantized into 8000 hypercubes with each state variable being quantized in 20 quantum levels. The code used was a binary, linearly independent code with elements +1 and -1. A set of training samples was obtained by generating optimal trajectories in the quantized state-space with several initial conditions, uniformly distributed in state space. The size of the training set was 600, the percentage being 7.5% of the total hypercubes.

A learning controller was simulated on digital computer, which was trained on the above mentioned training set. It was found that all the training samples were not classifiable by the controller. Mendel points out that this could be due to the switching surface being nonprojectable. It was found out that the percentage of correctly classified samples fluctuated around 93%.

The training was terminated after no more improvement was obtainable and after that the trained

controller was put to test on initial conditions not included in the training set. The controller was found to work satisfactorily.

The application of trainable controller up to date has been limited to two level controllers (i.e. bang-bang) only. The reason for this is due to the difficulties associated with obtaining training samples for multilevel controllers. Often two point boundary value problems must be solved; such problems become increasingly more difficult and computationally time consuming as the number of control levels increases.

Elaborate system simulation may be required in order to generate the necessary data for the training set. It has been suggested in an application study in which the feasibility of realizing a switching surface model for a man-in-the loop was investigated (Mendel [23]); to let man be in the loop to assist in obtaining training samples.

The problem of obtaining data for training is common to all pattern recognition problems. Often such data must be obtained as a part of a different and even larger study. For example in order to train a device to recognize lunar features, NASA designed and launched a special Surveyor spacecraft to obtain representative photographs of these features [29].

In ~~the~~ Mendel's work ~~described above,~~ the performance of the learning controller with one particular code was studied (i.e. with elements +1 and -1). To our knowledge no study has been yet made in order to investigate the convergence rate of training and the generalization characteristics of various linearly independent codes. We would like to point out in this connection that the choice of code largely determines the convergence rate of the training, as well as the generalization characteristics also depend on the code used. Another point regarding the code is the effect of coding quantum levels in different ways with a given code matrix.

It is not unusual to be unable to obtain a large training set. It is desired that the response of learning controller should be favourable even if a small training set is available and the training set belongs to a limited region of the state space.

The objectives of the present study are the following:

(i) to study how the choice of code affects the convergence rate of training and the generalization characteristics.

(ii) to study the behaviour of learning controller when quantum levels are coded in different ways with a given code matrix. Q.

(iii) the generalization characteristics of a learning controller for different codes when the training set belongs to a small region of the state space of interest.

(iv) to study a different learning algorithm called, "Relaxation Algorithm", and its convergence characteristics.

3.2 Introduction to Different Codes:

It has been pointed out by many neurologists that high reliability in brain functioning is achieved due to redundancy in neural networks. Every neuron has a large number of inputs or synapses (as large as 100,000) and a particular neuron does not "fire" unless there are a large number of "live" inputs. It has been noted that even if some neurons are destroyed or damaged, still the brain works fairly well. It would be interesting to use this idea of redundancy in case of learning machines, i.e., if a particular stimulus activates a large number of weights instead of a few, then during training the relevant information about the training sample would be stored in a distributed form in weights, and thus failure of a few weights would not destroy the "organization" of the machine. It is suggested here that this could be achieved by using codes having redundant structure.

In the present work, we have studied three types of codes. The code matrices have appearance as shown in Fig. 3.1, 3.2 and 3.3. The code matrices have rows which would yield linearly independent codes, since the rows become linearly independent by increasing dimension by addition of a +1 element to each row (corresponding to the threshold weight). It will be noticed that the rows from the code matrices of Fig. 3.2 and Fig. 3.3 have two distinct parts of different character (i.e. 0, 1 and -1, +1 respectively). Note too that the rows from the code matrix of Fig. 3.1 have only one element which has a different character (i.e. 1 as compared to zeros). We shall call for ease of notation the codes obtained from matrices of Fig. 3.2 and Fig. 3.3 as "01 contrast" and "$\pm$1 contrast" type codes respectively. The code obtained from the matrix of Fig. 3.1 exhibits very little property of "contrast". Since its form is very similar to an identity matrix, we shall call the code so obtained as "identity" type code.

The codes described above are said to be in normal form. It will be noticed that there is a regular change of order in rows as we move from the top row to the bottom row of any code matrix. In other words any row is very similar to the neighbouring rows. However if the rows of a normal code matrix are shuffled in a random manner so

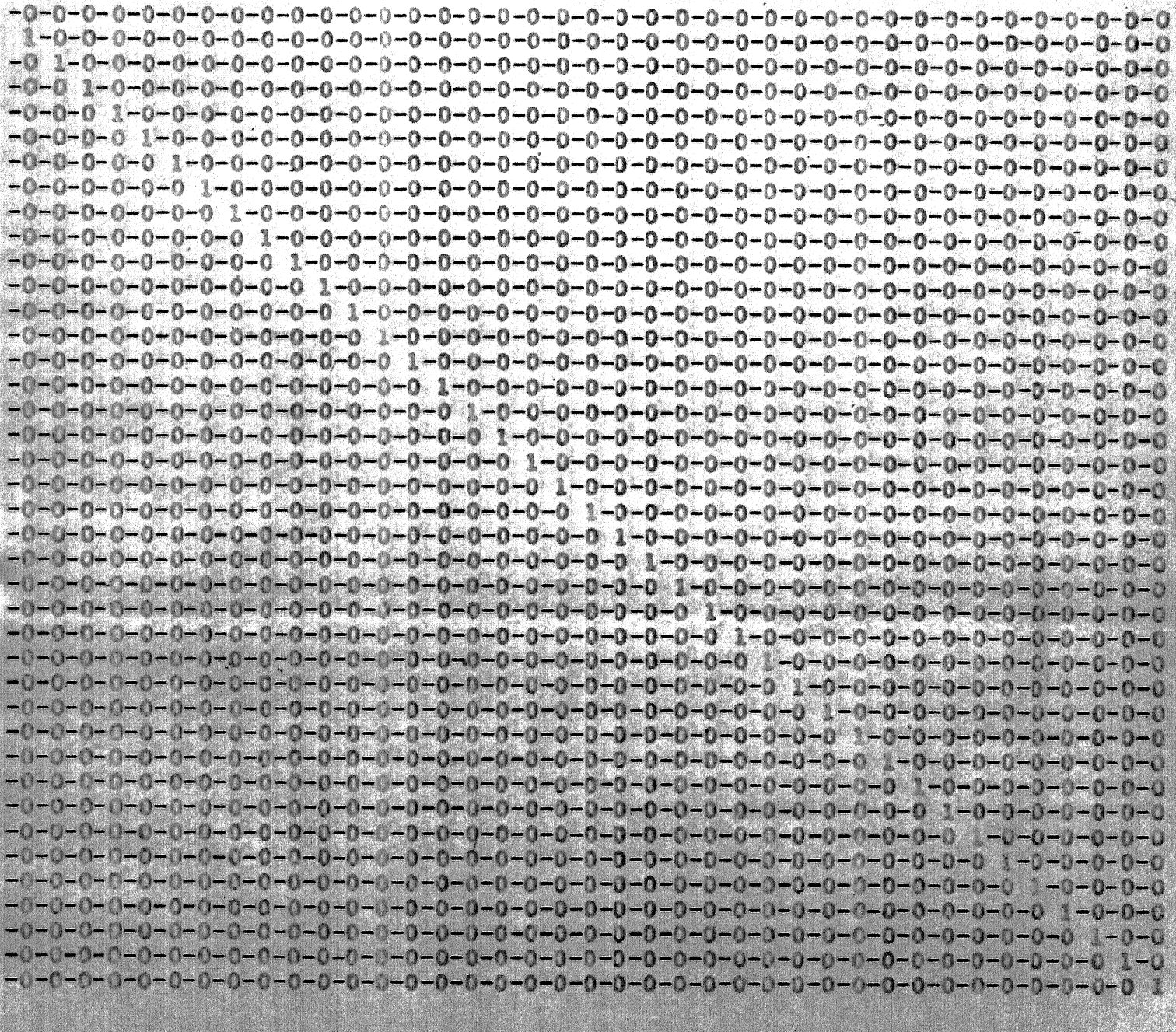


Fig. 3.1: A normal "identity" code matrix.

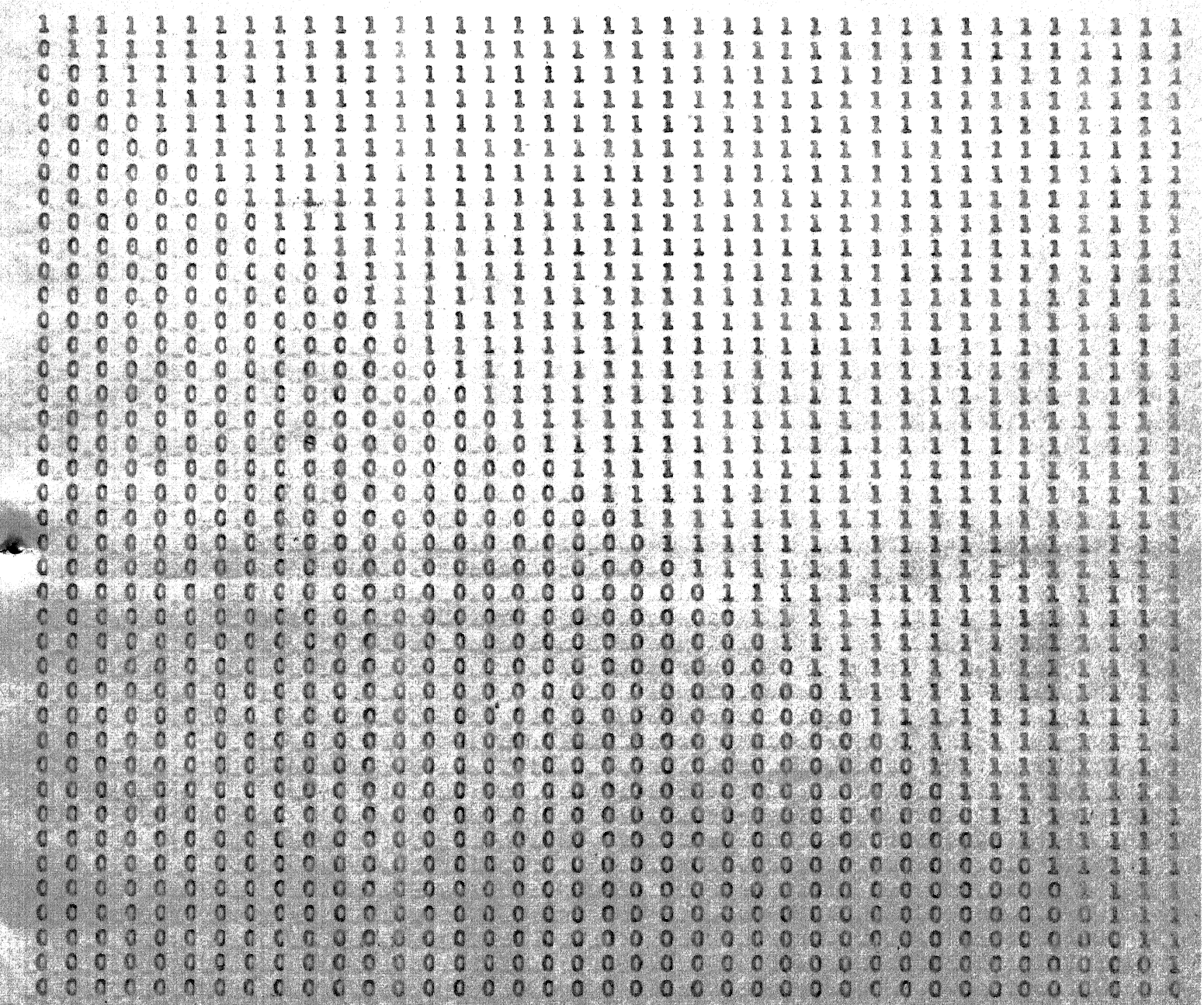


Fig. 3.2: A normal "01 contrast" code matrix.

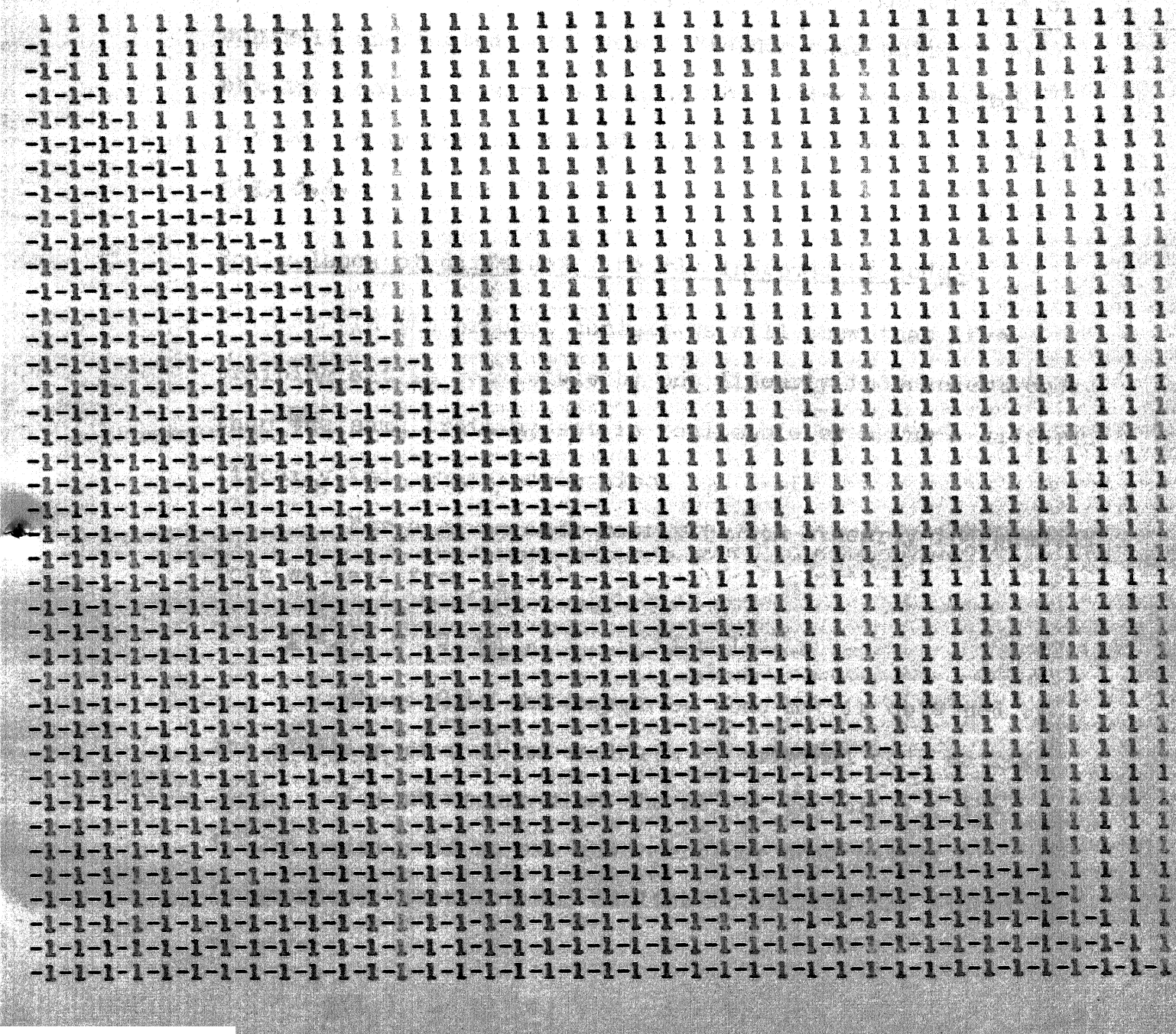


Fig. 3.3: A normal "±1 contrast" code matrix.

as to change the order of rows, the linear independence of codes is unaffected, and thus a "random code matrix" so obtained could be very well used for coding the quantum levels. A typical random ± 1 contrast type code is shown in Fig. 3.4.

Equivalence of different linearly independent codes:

In the present context we will show that if a training set is realizable by one linearly independent code, then the same training set is realizable by using a different linearly independent code also.

For a given code matrix Q with linearly independent rows we have from (2.17)

$$\underline{f}(X_i) = Q \underline{w}_i , \qquad i = 1,\ldots n \tag{2.17}$$

where Q is the augmented code matrix obtained by the addition of a +1 to each row of original code matrix.

Let us assume that the weight vectors $\underline{w}_i$, $i = 1,\ldots n$ exist. If we are given another code matrix Q' (augmented), then for the Adaline to realize the dichotomy we have

$$\underline{f}(X_i) = Q' \underline{w}'_i , \qquad i = 1,\ldots n \tag{3.2}$$

where $\underline{w}'_i$ are the new weight vectors.

or,

$$Q' \underline{w}'_i = Q \underline{w}_i , \qquad i = 1,\ldots n \tag{3.3}$$

Fig. 3.4: A typical random ±1 contrast code matrix.

or,

$$\underline{w}_i' = \left[Q'\right]^{-1} Q\, \underline{w}_i , \qquad i = 1,\ldots n \tag{3.4}$$

Thus if the new code matrix has linearly independent rows then a set of corresponding weight vectors $\underline{w}_i'$ exists and the same dichotomy can be realized.

Effect of threshold:

The addition of a+1 element to each row of a code matrix has a special significance. This not only ensures the linear independence of codes but also serves the following purposes.

(i) In any physical realization of an Adaline, the quantity $\underline{W}^T\underline{S}$ in (2.3) will be noisy, and probably the system for detecting the sign of $\underline{W}^T\underline{S}$ will not give repeatable performance for values of $\underline{W}^T\underline{S}$ near zero. For these reasons it is desirable to find a weight vector $\underline{W}$ such that the magnitude of $\underline{W}^T\underline{S}$ is greater than some minimum value. By the introduction of a threshold weight it is possible to adjust it so as to give a satisfactory performance.

(ii) As we know, by quantization-encoding a mapping is achieved from state space onto the vertices of a unit kq dimensional cube in the pattern space. A situation could occur, similar to that shown in Fig. 3.5, for a two dimensional pattern space.

It will be noticed that although the points of the two classes are separable by means of a straight line, the separating line does not pass through origin, and thus an equation of type $y = mx$ will not satisfy the requirements of partition. It will be noticed that if we add an extra dimension Z and move each point in the XY plane by a distance +1 along this dimension, then we can find a plane which will separate the samples of the two classes, the plane passing through origin (this being equivalent to adding a new weight i.e. threshold weight and a corresponding element +1 to each stimulus). The search of such a separating plane will always be successful since from Euclid's theorem we have:

Given a straight line and a point in space, there exists a plane which contains the straight line and passes through the given point.

Thus we find that introduction of the threshold endows flexibility to the recognition task.

Generalization

The function of a learning controller is to define the complete control law for a plant, with the knowledge of the correct control only at a finite number of points. This means that the trained controller is required to provide an implicit definition of the entire control law

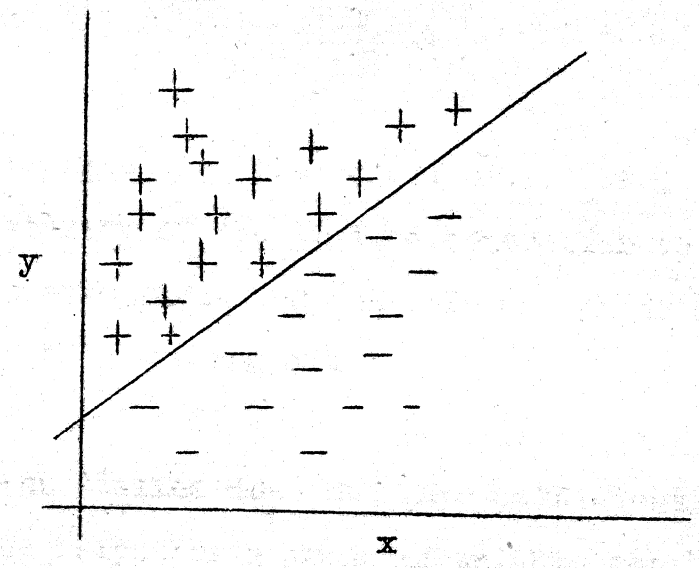


Fig. 3.5 A partitioning dichotomy in two dimensions.

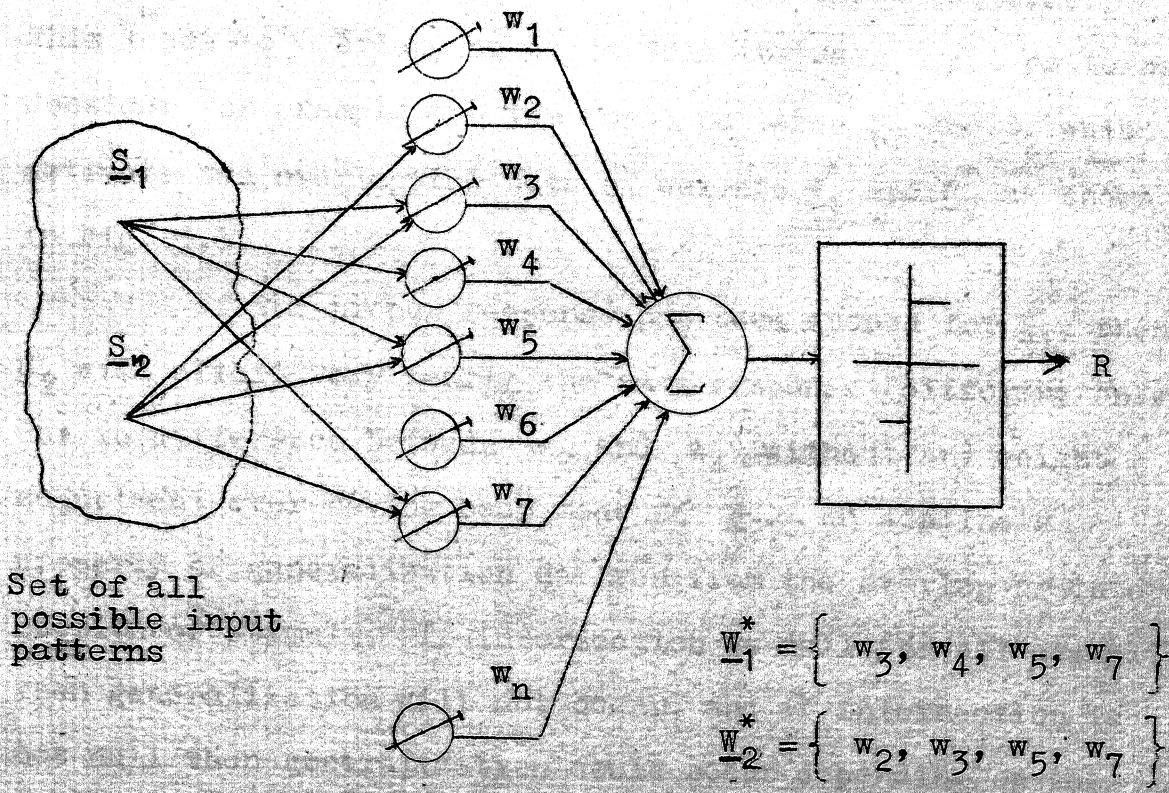


Fig. 3.6 Intersection of input stimuli in an Adaline.

However if it is desired that the input stimulus $\underline{S}_2$ produce a markedly different response from $\underline{S}_1$, then the overlap of stimuli might create difficulty. Adjustment of all the weights attached to $\underline{S}_2$ so as to produce the proper response for $\underline{S}_2$ will upset most of the weights which contribute to the output for $\underline{S}_1$. This is called learning interference, and is similar to retroactive inhibition experienced by biological organisms when presented with highly similar stimuli for which different responses are required.

Learning interference can be overcome by repeated application of a learning algorithm. Repeated adaptations eventually lead to sufficiently large magnitudes being attatched to the few weights which do not correspond to the overlap of the stimuli. In summary the Adaline's ability to generalize is due to the overlap of stimuli. Thus from the above discussion we expect the Adaline to generalize better with contrast type codes as compared to identity type codes.

Smith [24] has shown that generalization is ensured after training on the samples on the boarder of the decision boundary if the following conditions have been satisfied.

(i) There are no cross product terms in the expression for the switching surface.

(ii) The switching surface is monotonic in one or more of the state variables.

Smith also indicates that the Adaline has a strong tendency to generalize correctly even if the above conditions are not met.

No formalization of generalization criteria exists beyond that stated above. The usual approach to verify the presence of generalization in a practical problem is to test it by using samples not included in the original training set.

3.3 Learning Algorithms

The learning algorithm associated with a single daline, adjusts the weight vector W to realize a solution f (2.3). Several methods of adjusting the weights exist as etailed.

Let A and B denote two classes. An output of +1 rom Adaline means that the input stimulus is recognized as member of A and on the contrary an output of -1 means that he input stimulus is recognized as a member of B.

One method of adjusting the weightsis the forced-earning-method. In this method each input stimulus element s added to the corresponding weight if the stimulus is a mber of A, and is subtracted if the input is a member of This method of learning has the disadvantage of being pendent on the sequence of training samples and of not aranteeing a solut

Second method of weight adjustment is the error-correction-method. In this method the weights are not changed if the Adaline classifies a stimulus correctly. If the input pattern is actually a member of A but Adaline classifies it as a member of B, then each input stimulus element multiplied by a constant η is added to the corresponding weights. If the input stimulus is actually in B but is classified as a member of A, then each stimulus element multiplied by a constant η is subtracted from the corresponding weights. For correction of any error, η remains at a certain positive value. Generally the value of η is taken to be 1. In the present work we have considered this algorithm for $\eta = 1$. Error correction guarantees a solution for (2.3) if η is chosen to satisfy certain conditions and the training set is linearly separable. The proof of this method can be found in Nilsson[26], Block [34] and Rosenblatt [31].

Another method of adjusting the weights is the Widrow-Hoff-minimum-mean-square algorithm. In this algorithm, the error for an input is defined as the desired output minus the weighted sum of stimulus elements. At each presentation of an input stimulus, each stimulus element times the error times a constant is added to the corresponding weights. It has been shown that if the mean square error is a hyperparabolic function of the weights, then this algorithm

is equivalent to searching the minimum of the hyperparabola using the method of steepest descent. This algorithm tends to cluster all input sums for members of A, around +1 and all input sums for members of B around -1. It is possible to modify this algorithm so that the weights are adjusted only if the Adaline output is wrong (i.e. error correction).

Another method of changing weights is the relaxation algorithm. This algorithm is stated mathematically as follows.

Let $\underline{W}(k)$ be the present weight vector. If an error occurs, the weight vector after correction will be $\underline{W}(k+1)$.

The relaxation rule is

$$\underline{W}(k+1) = \underline{W}(k) - \lambda \frac{\underline{W}^T(k)\,\underline{S}(k)}{\|\underline{S}(k)\|^2}\,\underline{S}(k)\,,$$

$$\text{if} \quad \underline{W}^T(k)\,\underline{S}(k) \leq 0 \qquad (3.5)$$

and

$$\underline{W}(k+1) = \underline{W}(k)\,, \qquad \text{if} \quad \underline{W}^T(k)\,\underline{S}(k) > 0$$

where $\underline{S}(k)$ is a row of A in (2.12).

This algorithm has a simple geometric meaning. The quantity,

$$r_k = -\frac{\underline{W}^T(k)\,\underline{S}(k)}{\|\underline{S}(k)\|} \qquad (3.6)$$

is the distance from the tip of vector $\underline{W}(k)$ to the hyperplane $\underline{W}^T(k)\ \underline{S}(k) = 0$. Since $\underline{S}(k)/\|\underline{S}(k)\|$ is the unit normal vector for that hyperplane, (3.5) calls for $\underline{W}(k)$ to be moved a certain fraction λ of the distance from $\underline{W}(k)$ to the hyperplane. If $\lambda = 1$, $\underline{W}(k)$ is moved exactly on the hyperplane. The convergence of the above algorithm is guaranteed for

$$0 < \lambda \leq 2$$

for a linearly separable training set. The algorithm may be convergent for $\lambda > 2$, but convergence is not guaranteed. The proof is as follows.

Let $\underline{W}^*$ be a solution weight vector. Thus if the algorithm is convergent then with each correction the weight vector $\underline{W}(k)$ must get closer to $\underline{W}^*$. Since

$$\underline{W}(k+1) = \underline{W}(k) - \lambda \frac{\underline{W}^T(k)\ \underline{S}(k)}{\|\underline{S}(k)\|^2}\ \underline{S}(k)$$

we have

$$\|\underline{W}(k+1) - \underline{W}^*\|^2 = \underline{W}^T(k)\ \underline{W}(k) + \lambda^2\left[\frac{\underline{W}^T(k)\ \underline{S}(k)}{\|\underline{S}(k)\|^2}\right]^2 \underline{S}^T(k)\ \underline{S}(k)$$

$$- 2\lambda\left[\frac{\underline{W}^T(k).\underline{S}(k)}{\|\underline{S}(k)\|^2}\right]\underline{W}^T(k)\ \underline{S}(k) + \underline{W}^{*T}\ \underline{W}^* - 2\underline{W}^T(k)\ \underline{W}^*$$

$$+ 2\lambda\left[\frac{\underline{W}^T(k).\underline{S}(k)}{\|\underline{S}(k)\|^2}\right]\underline{W}^{T*}\ \underline{S}(k) \qquad (3.7)$$

or

$$\| \underline{W}(k+1) - \underline{W}^* \|^2 = \| \underline{W}(k) - \underline{W}^* \|^2 + 2\lambda \frac{\underline{W}^T(k)\, \underline{S}(k)}{\| \underline{S}(k) \|^2} \cdot$$

$$\left[\underline{W}^* - \underline{W}(k)\right]^T \underline{S}(k) + \lambda^2 \left[\frac{\underline{W}^T(k)\, \underline{S}(k)}{\| \underline{S}(k) \|^2}\right]^2 \qquad (3.8)$$

Now since $\underline{W}^{*T}\underline{S}(k) > 0$ for any pattern $\underline{S}(k)$ from the training set, we have,

$$\left[\underline{W}^* - \underline{W}(k)\right]^T \underline{S}(k) > -\underline{W}(k)^T \underline{S}(k) \geq 0 \qquad (3.9)$$

from (3.9) and (3.8) we have

$$\| \underline{W}_{k+1} - \underline{W}^* \|^2 \leq \| \underline{W}_k - \underline{W} \|^2 - \lambda (2 - \lambda) \frac{\left[\underline{W}(k)^T\, \underline{S}(k)\right]^2}{\| \underline{S}(k) \|^2} \qquad (3.10)$$

Thus from (3.10) we find that $\underline{W}(k)$ gets closer to $\underline{W}^*(k)$ with each correction as long as λ is restricted to the range $0 < \lambda \leq 2$. In other words the sequence of weight vectors $\underline{W}(1)$, $\underline{W}(2)$... gets closer and closer to $\underline{W}^*$ and in the limit as k goes to a large value the distance $\| \underline{W}(k) - \underline{W}^* \|$ approaches some limiting distance $r(\underline{W}^*)$. This means that as k goes to infinity $\underline{W}(k)$ is confined to the surface of a hypersphere with center at $\underline{W}^*$ and radius $r(\underline{W}^*)$. Since this is valid for any solution weight vector $\underline{W}^*$, the limiting $\underline{W}(k)$ is confined to the intersection of the hyperspheres centered about all the possible solution vectors.

It can be shown that the common intersection of these hyperspheres is a single point on the boundary to the solution region [25], [26], [27].

This algorithm has been used in automatic classification of chemical compounds by Isenhour and Jurs [28], for $\lambda = 2$.

3.4 Learning with Two Levels of Control: Time Optimal Learning Control System:

As noted earlier, an Adaline can dichotomize the feature space after being trained on a training set, the Adaline can readily be used as a time optimal controller as its response is bang-bang type.

We shall first study the performance of learning controller for two dimensional case i.e. with two state variables, as it is easy to study the generalization characteristics in two dimensions.

Before studying the actual time optimal control problem we shall briefly examine how an Adaline generalizes after being trained on a training set of two samples for different codes.

A learning controller was simulated for learning two training samples using $\pm(1)$ contrast, 01 contrast and identity codes for the two dimensional state space. Each state was quantized into 40 quantum levels. A code was

assigned to each quantum level, coding being done with the normal code matrices of Fig. 3.1, 3.2 and 3.3. The learning algorithm used was fixed increment learning method and initial weight vector was arbitrarily chosen to be zero.

Fig. 3.7, 3.8 and 3.9 show the generalization maps of the controller after being trained on the training samples marked. It will be noticed from these figures that the Adaline coded with the contrast type codes, responds to "unseen" patterns on the basis of having "seen" similar patterns during training. As noted earlier this is due to the effect of overlap or intersection of patterns. On the other hand the Adaline coded with identity type code seems practically inert to new samples.

From the above discussion it is clear that the controller using contrast type codes has a tendency to generalize over a large space. This property is particularly useful when training set is small and the training samples are widely separated.

Let us now consider the learning and generalization characteristics for a larger set of training samples with fixed increment learning algorithm. An arbitrary time optimal switching surface was chosen as shown in Fig. 3.10. The states (X,Y) were quantized in 40 quantum levels each. A set of fifty training samples was chosen for training the controller as shown in Table I.

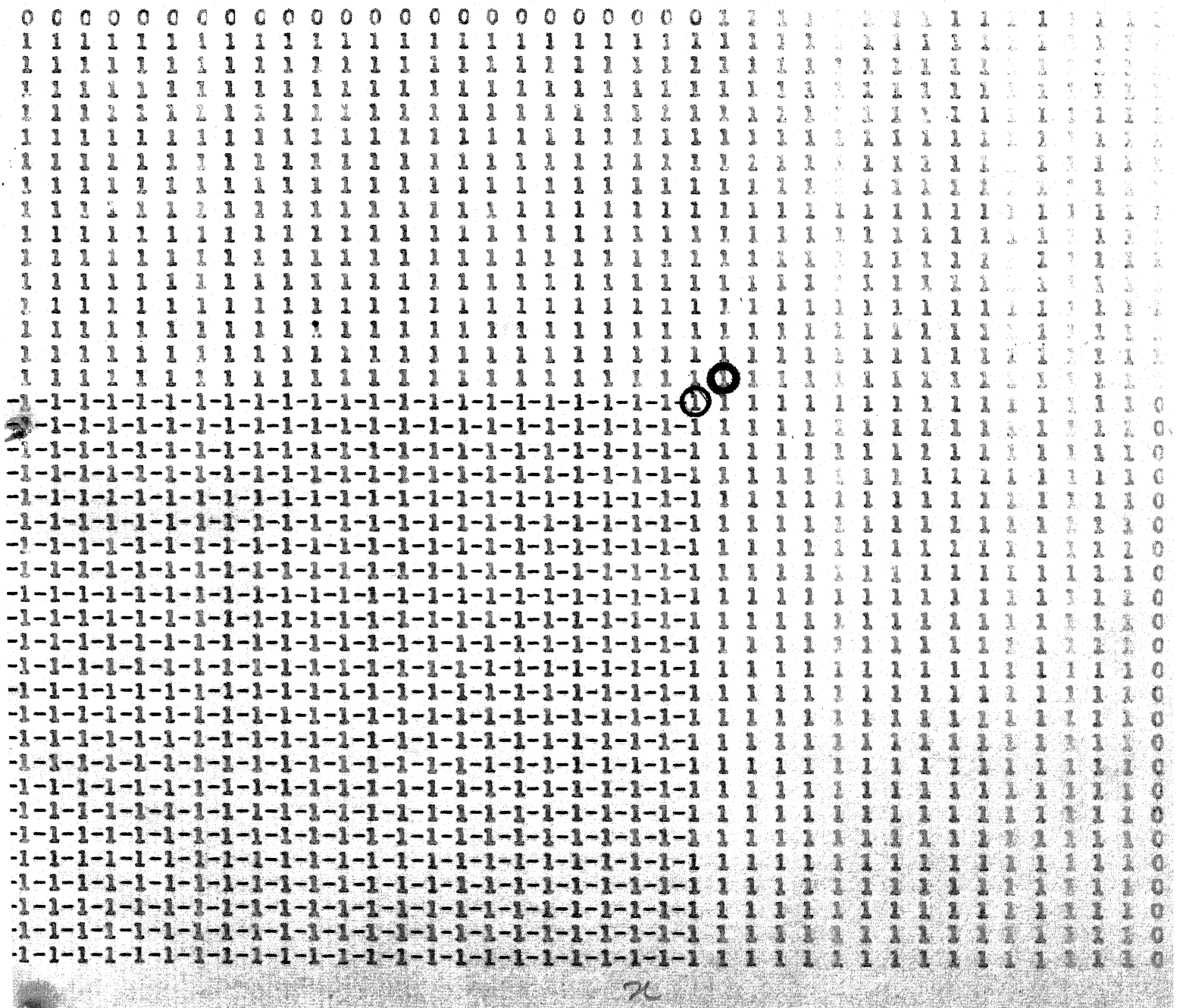


Fig. 3.8: Generalization map of the controller trained on two training samples with normal 01 contrast code. Marked locations indicate training samples.

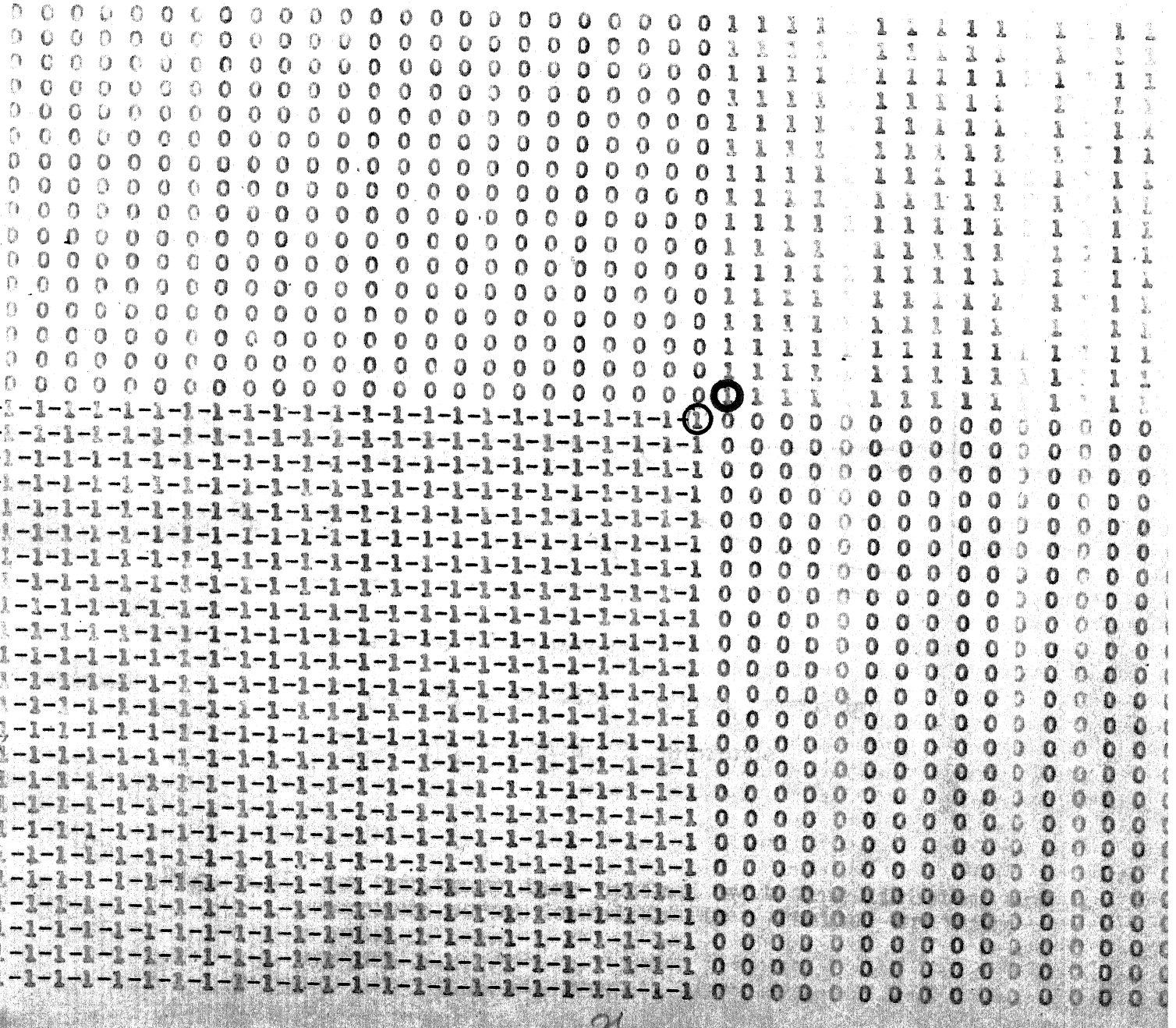
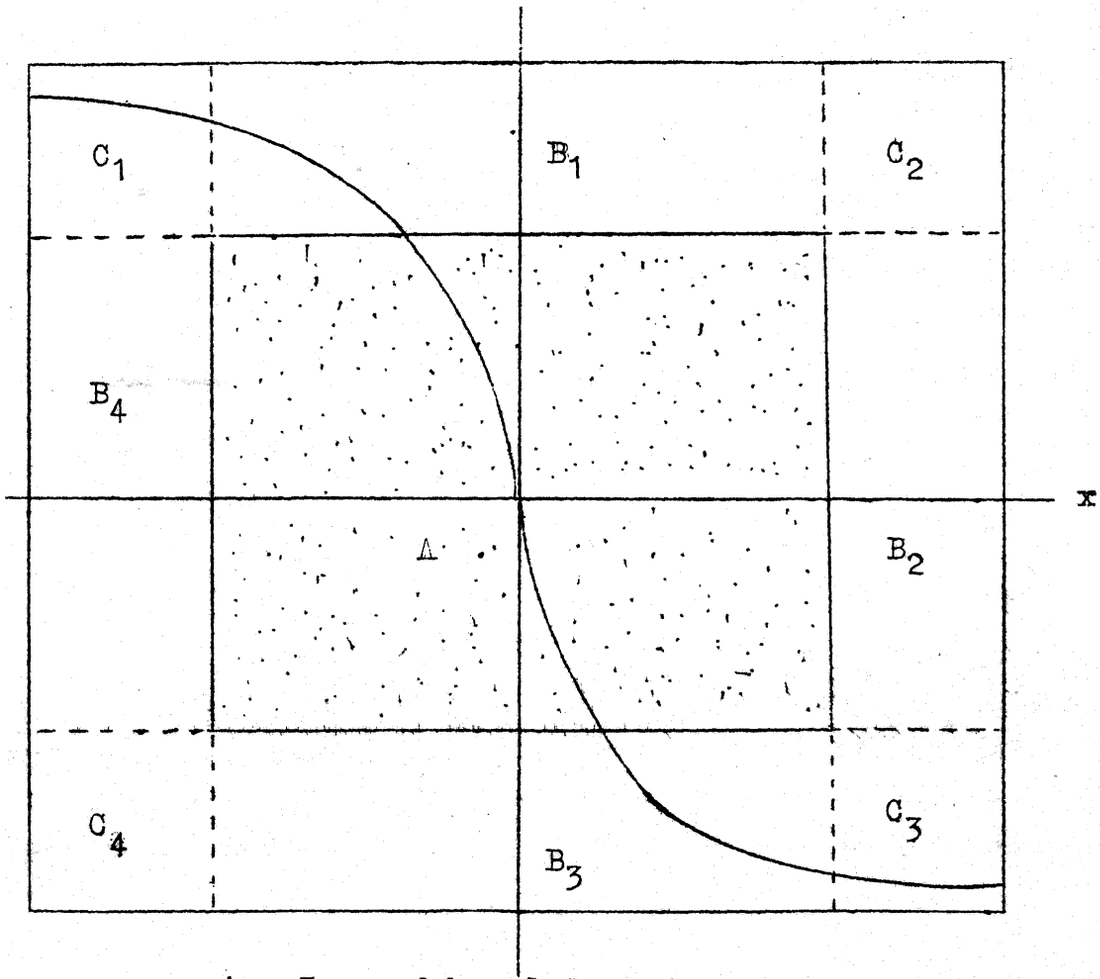


Fig. 3.9: Generalization map of the controller trained on two training samples with normal ±1 contrast code. Marked locations indicate the training samples.



A = Zone of knowledge

$B_1,\ldots,B_4$ = Zones of partial knowledge

$C_1,\ldots,C_4$ = Zones of ignorance

Fig. 3.10 An arbitrary time optimal switching dichotomy and various zones formed due to a limited training set.

Table I

A training set for training a time optimal learning controller

S. No.	X axis quantum number	Y axis quantum number	Control	S. No.	X axis quantum number	Y axis quantum number	Control
1	26	12	+1	26	30	19	-1
2	10	32	-1	27	12	29	+1
3	12	31	-1	28	14	26	+1
4	14	30	-1	29	14	28	+1
5	22	30	-1	30	17	26	+1
6	15	29	-1	31	14	24	+1
7	17	28	-1	32	18	24	+1
8	24	28	-1	33	14	22	+1
9	19	26	-1	34	18	22	+1
10	24	26	-1	35	16	21	+1
11	20	25	-1	36	20	21	+1
12	26	23	-1	37	16	19	+1
13	26	20	-1	38	20	19	+1
14	22	20	-1	39	16	17	+1
15	28	18	-1	40	20	17	+1
16	22	18	-1	41	22	15	+1
17	23	16	-1	42	20	14	+1
18	30	16	-1	43	24	13	+1
19	25	14	-1	44	20	12	+1
20	27	13	-1	45	22	12	+1
21	29	12	-1	46	31	9	+1
22	31	11	-1	47	21	26	-1
23	32	10	-1	48	29	10	+1
24	30	24	-1	49	14	11	+1
25	10	30	+1	50	26	10	+1

The training samples depicted in Table I, were chosen from the zone marked A in Fig. 3.10. The samples were chosen sufficiently close to the optimal switching curve. Control situations, through which the switching curve passed, were not taken as training samples.

An Adaline was simulated and was trained on the above training set for different codes with initial weight vector as zero. Training was achieved by presenting the training samples in the sequence as shown in Table I. The learning curves of the Adaline for different codes are shown in Fig. 3.11. It will be noticed that the a priori probability P_S of correct dichotomization rapidly increases and finally attains a value unity, indicating that the switching surface is projectable. It will be also noticed that the convergence for the Adaline using ± 1 contrast type code is rapid as compared to other codes. The learning curve follows (roughly) an exponential curve and P_S fluctuates about a value close to unity for large values of number of corrections and finally converges to unity. These fluctuations are observed in all experiments performed with this algorithm and seems to be related to the non-optimum movements of the partitioning hyperplane. Adaline needed 236 corrections with identity code, 588 corrections with 01 contrast code and 147 corrections with ± 1 contrast code. The generalization maps

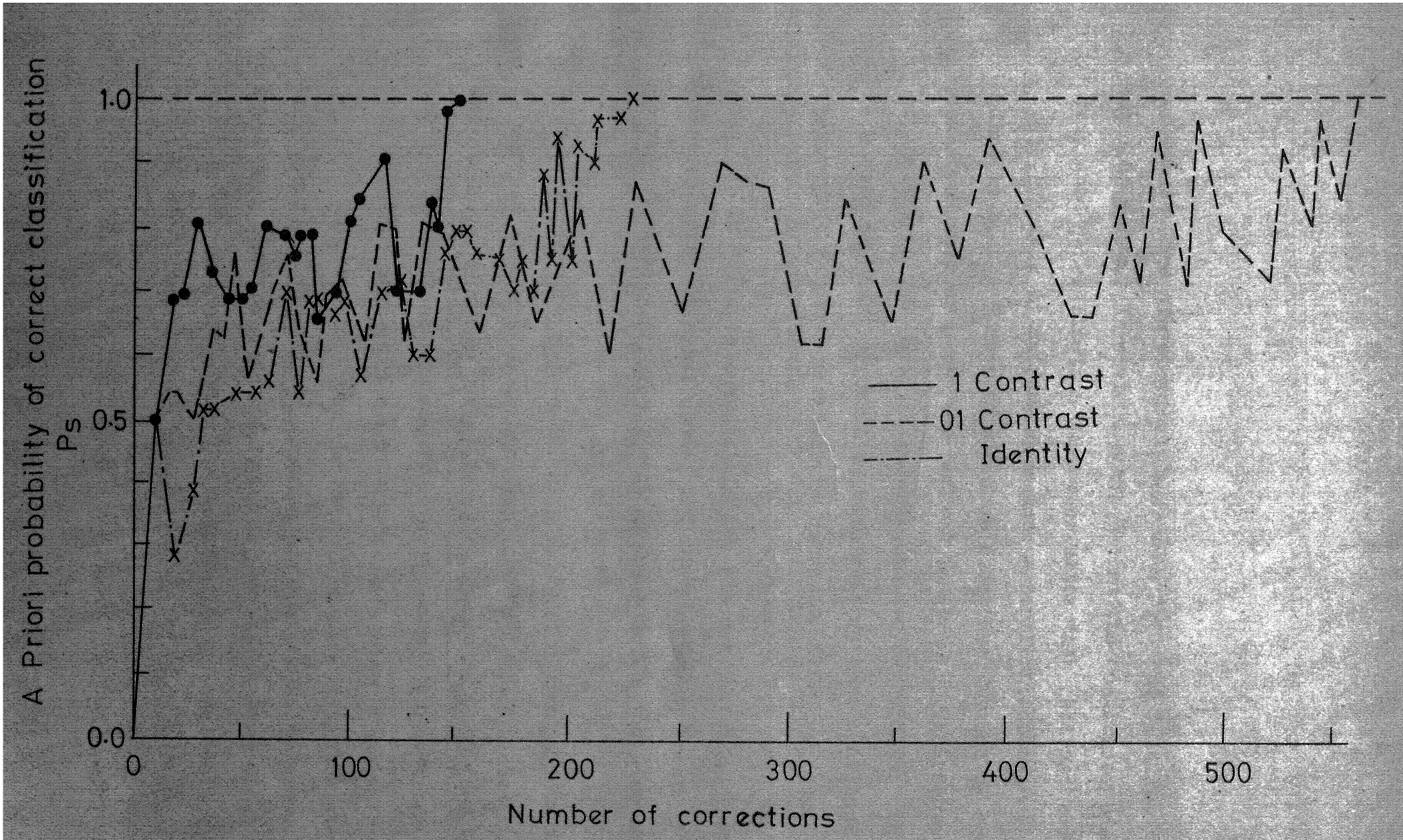


FIG. 3·11 TYPICAL LEARNING CURVES OF ADALINE WITH FIXED INCREMENT ERROR CORRECTION ALGORITHM

of the Adaline are shown in Fig. 3.12, 3.13 and 3.14 for various codes. It will be noticed that the controller has a very good generalization characteristics with contrast type codes, whereas it suffers from poor generalization with identity code. Number of misclassifications in case of contrast codes was very small in the zone of knowledge. The behaviour of Adaline in the zone of ignorance is not favourable. This is due to the lack of training samples from this zone, and improvement in performance could be achieved if some more training samples are included in the training set from this zone.

In order to investigate the generalization characteristics with different codes, the Adalines using random codes of Fig. 3.15(a) and 3.16(a) were trained on the training set of Table I, and tested for generalization. Fig. 3.15(b) and 3.16(b) depict the corresponding generalization maps. It will be noticed that the generalization is badly deteriorated. It was observed that the training took a larger number of iterations. The Adaline needed 543 corrections for ± 1 contrast code and 2306 corrections for 01 contrast code.

In order to study the generalization characteristics in greater detail another training set was chosen as shown in Table II. The training samples were now chosen from zones C_1 and C_3. The generalization characteristics of the controller for the three normal codes appear in Fig. 3.17, 3.18 and 3.19.

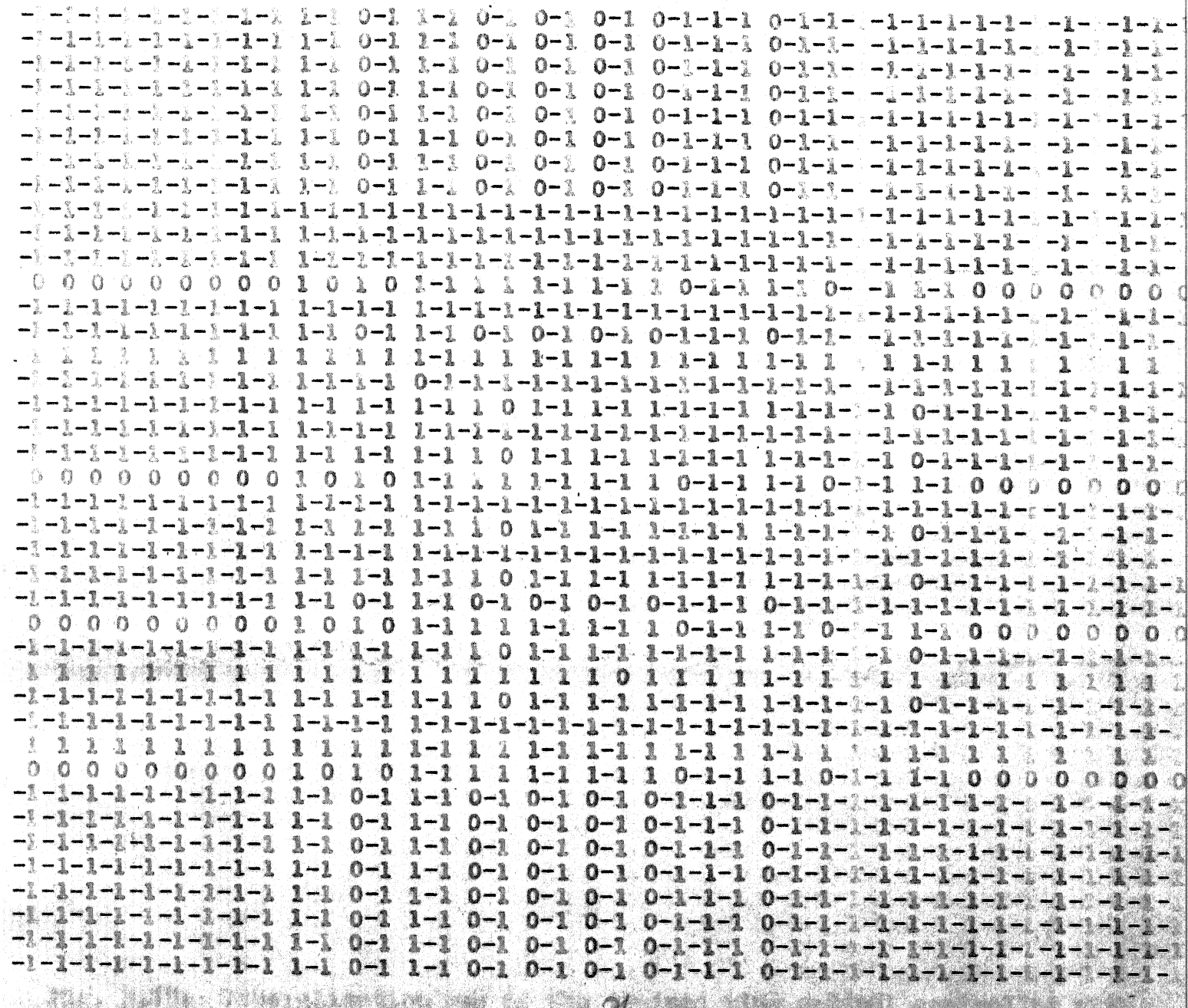


Fig. 3.12: Generalization map of the trained time optimal controller with normal identity code. (Training set of Table I).

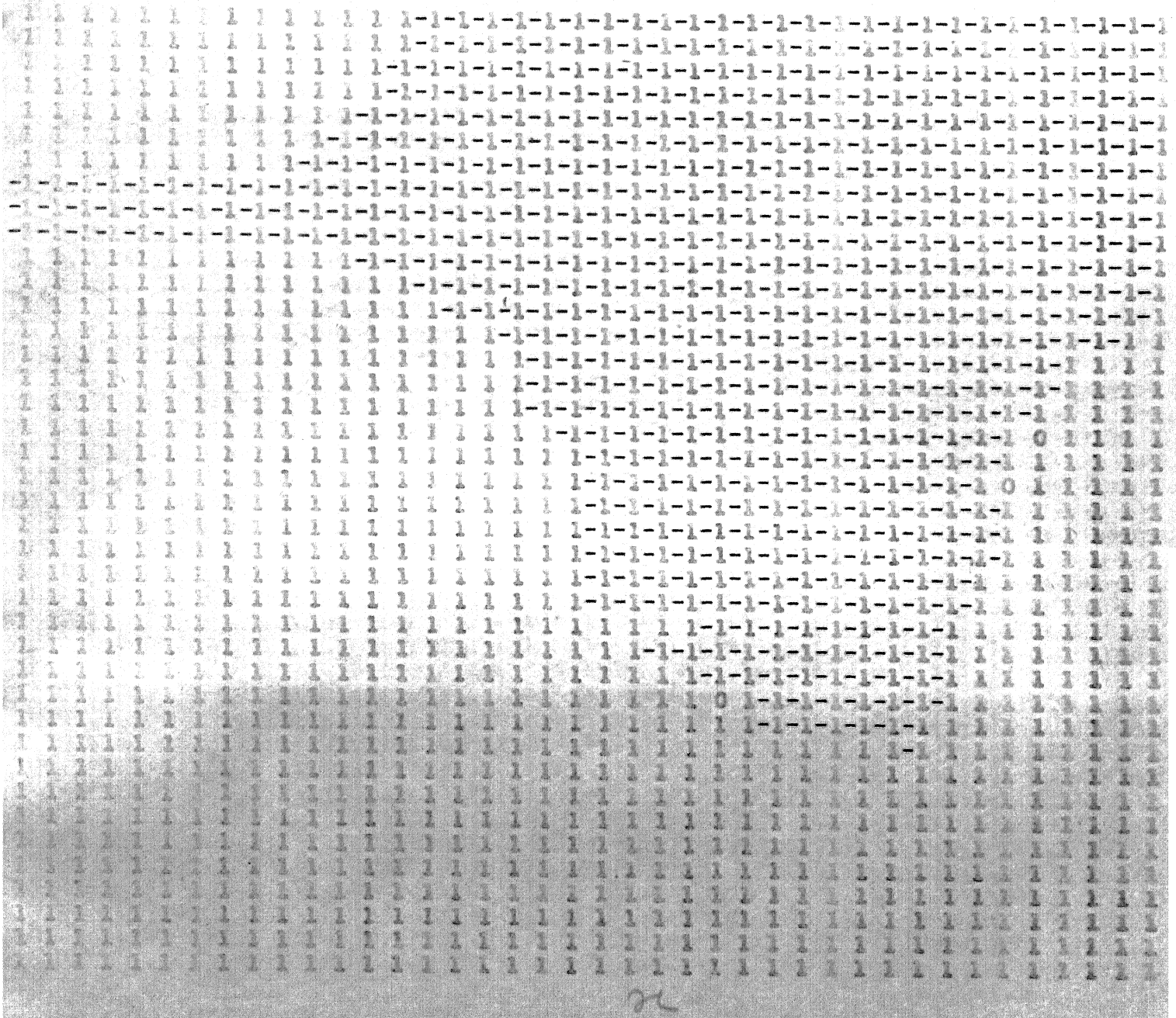


Fig. 3.13: Generalization map of the trained time optimal controller with normal 01 contrast code. (Training set of Table I).

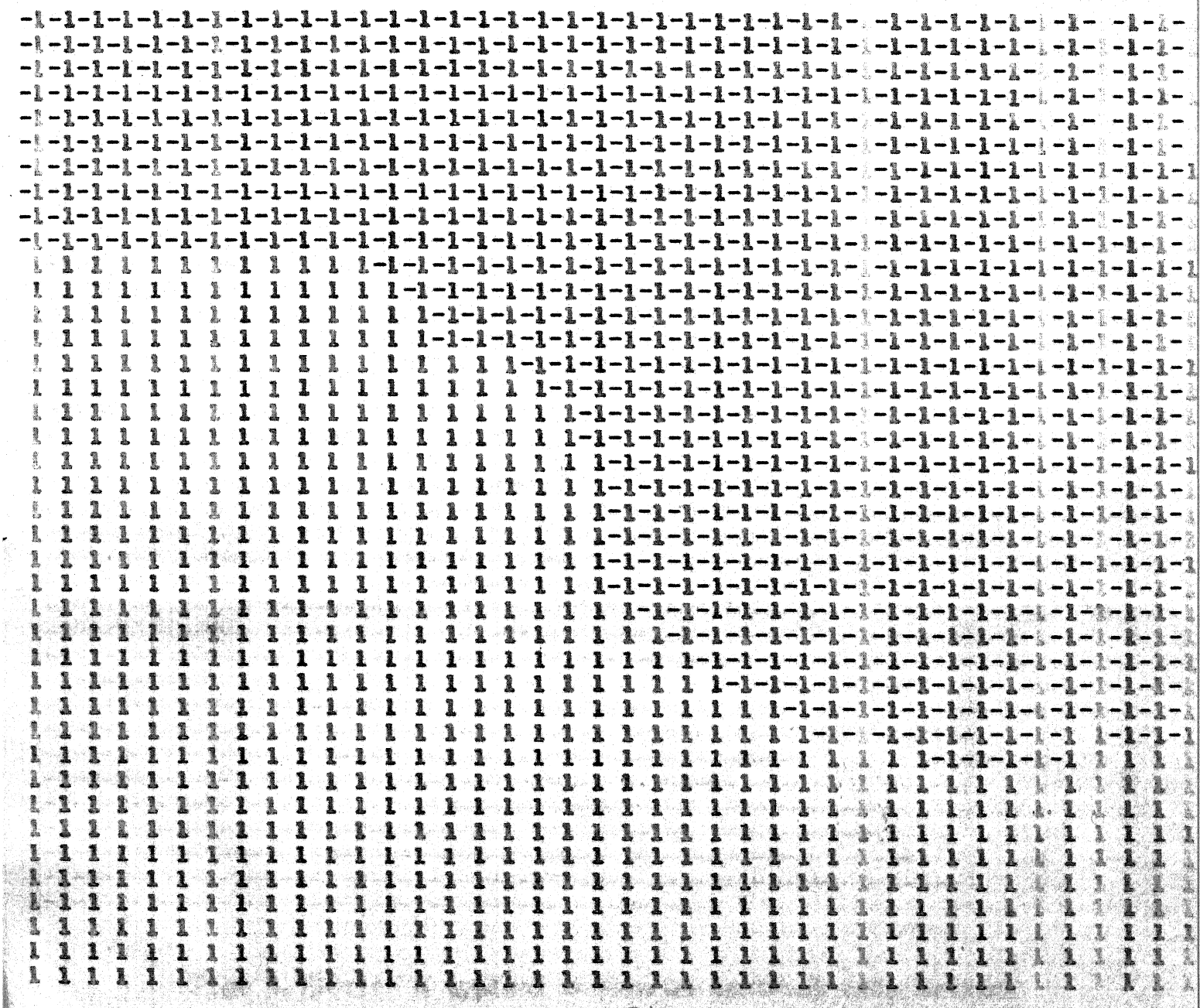


Fig. 3.14: Generalization map of the trained time optimal controller with normal ±1 contrast code. (Training set of Table I).

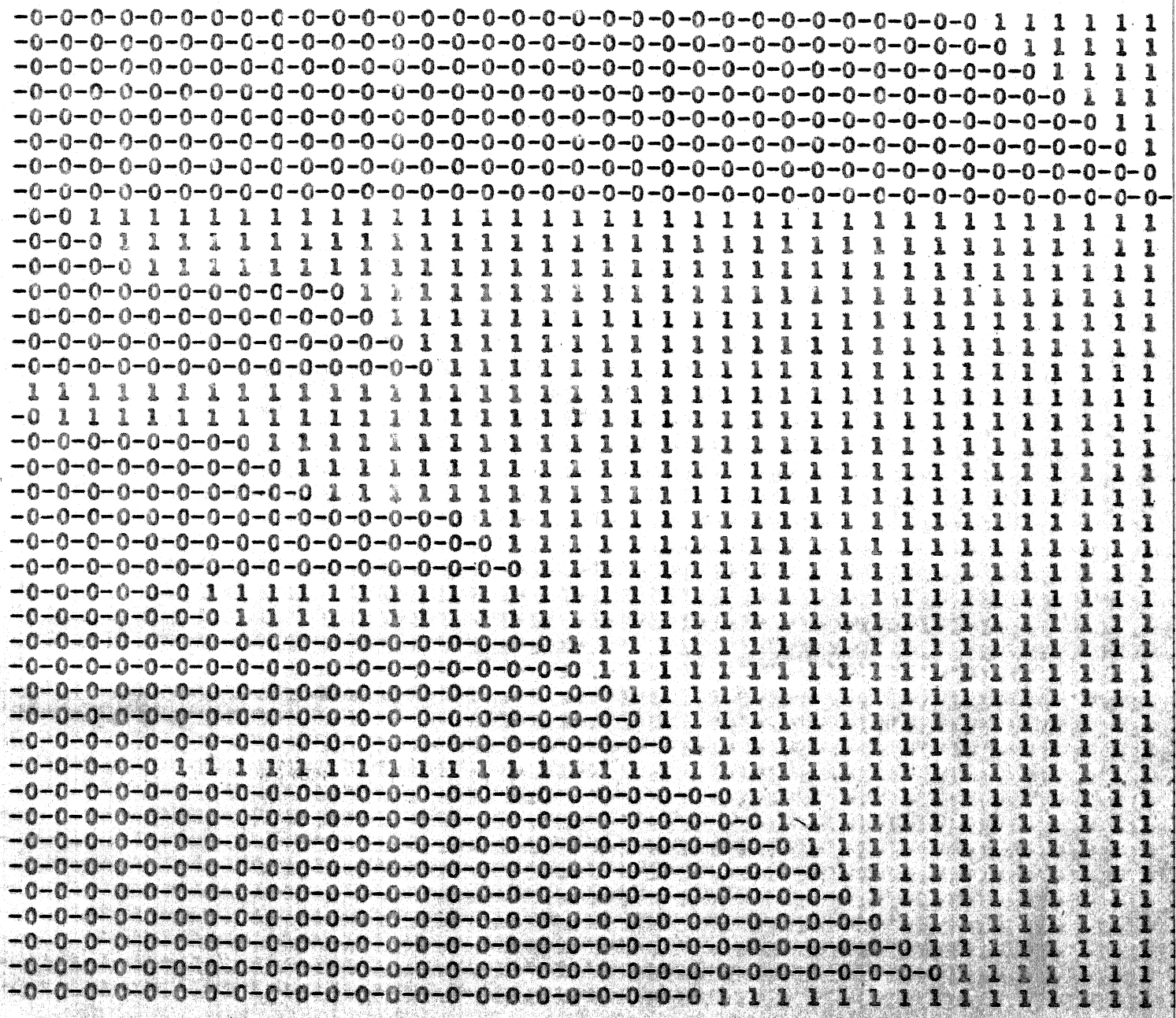


Fig. 3.15(a): A typical random 01 contrast code matrix.

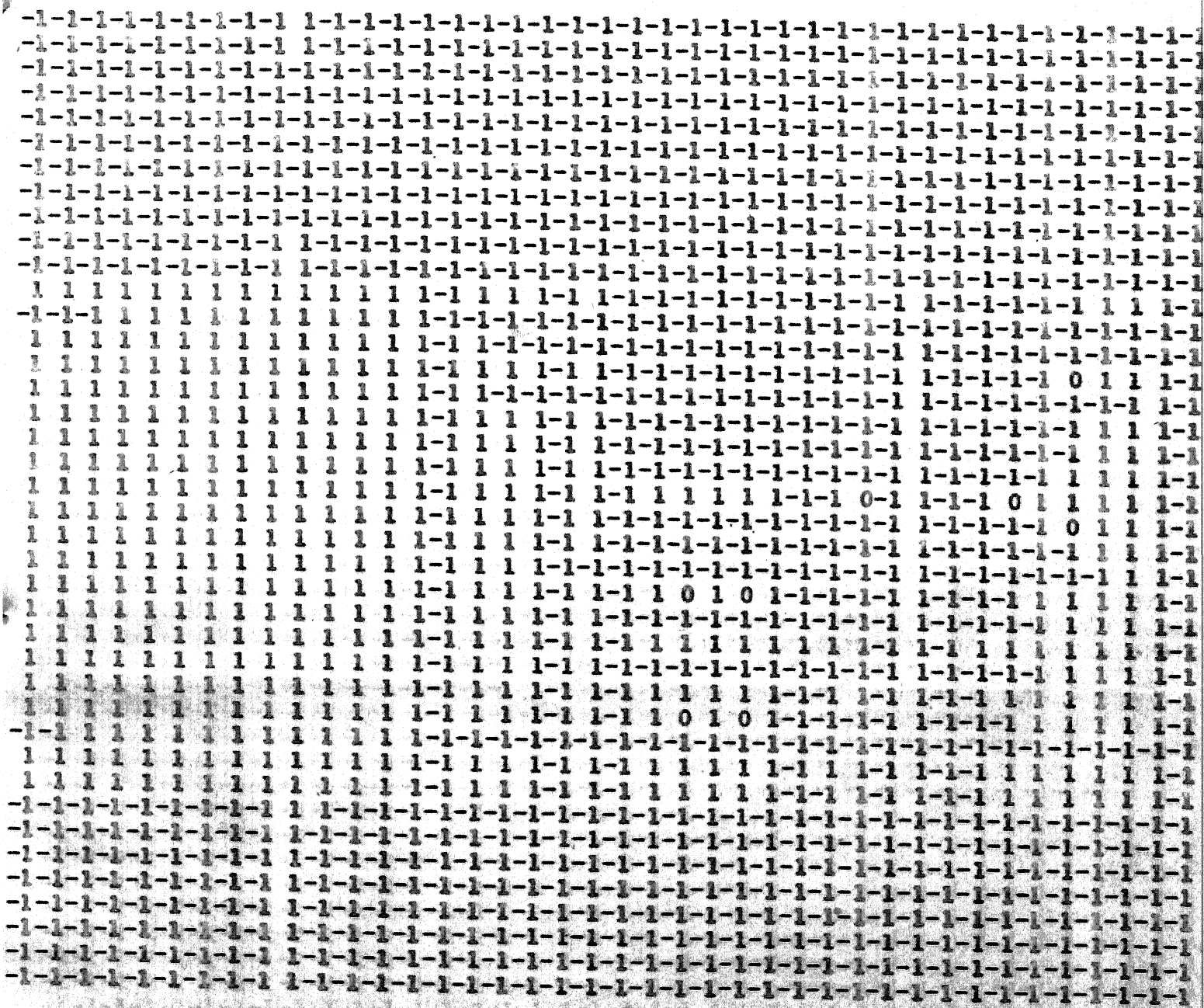


Fig. 3.15(b): Generalization map of the trained time optimal controller with random code matrix of Fig. 3.15(a). (Training set of Table I).

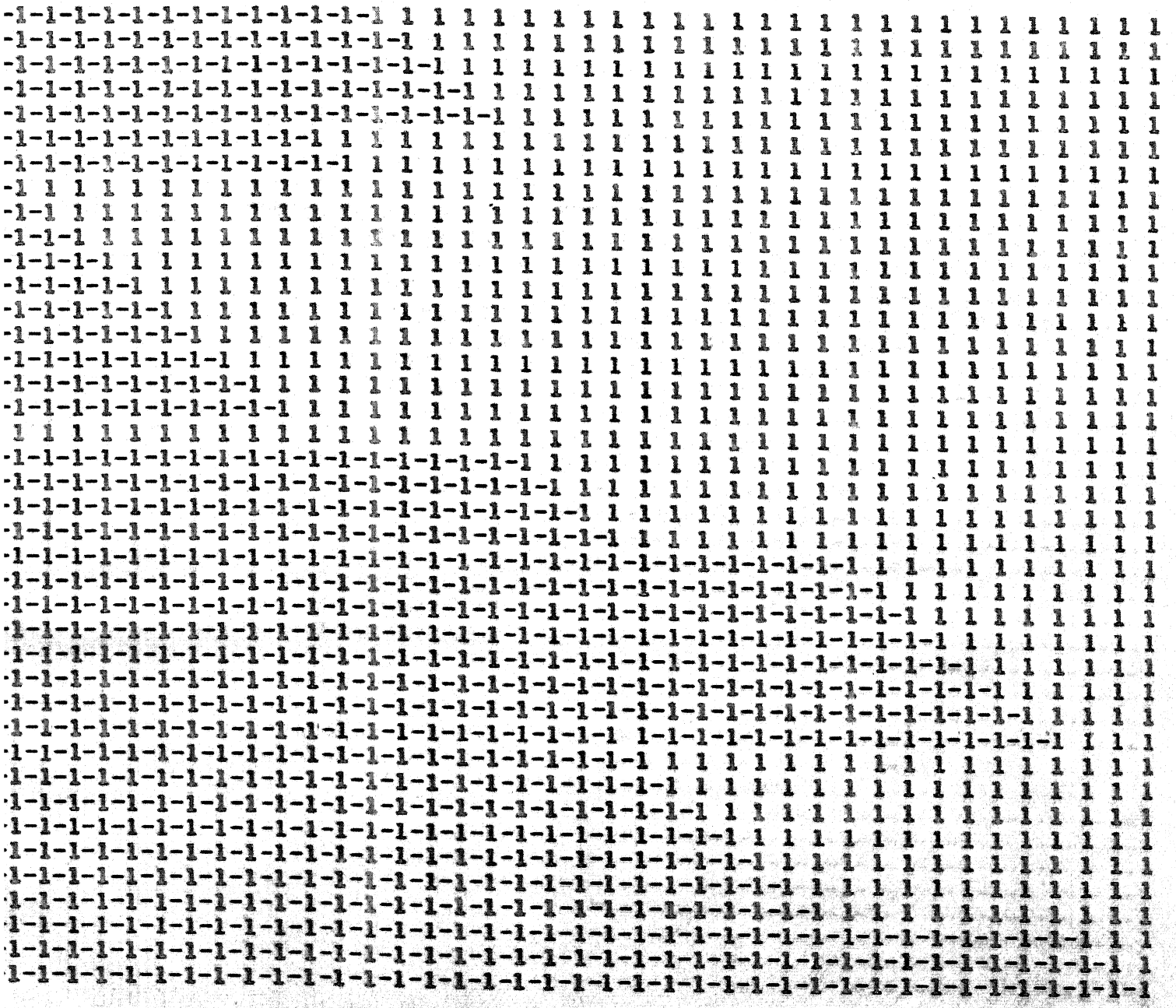


Fig. 3.16(a): A typical random ±1 contrast code matrix.

Fig. 3.17: Generalization map of trained time optimal controller with normal identity code. (Training set of Table II).

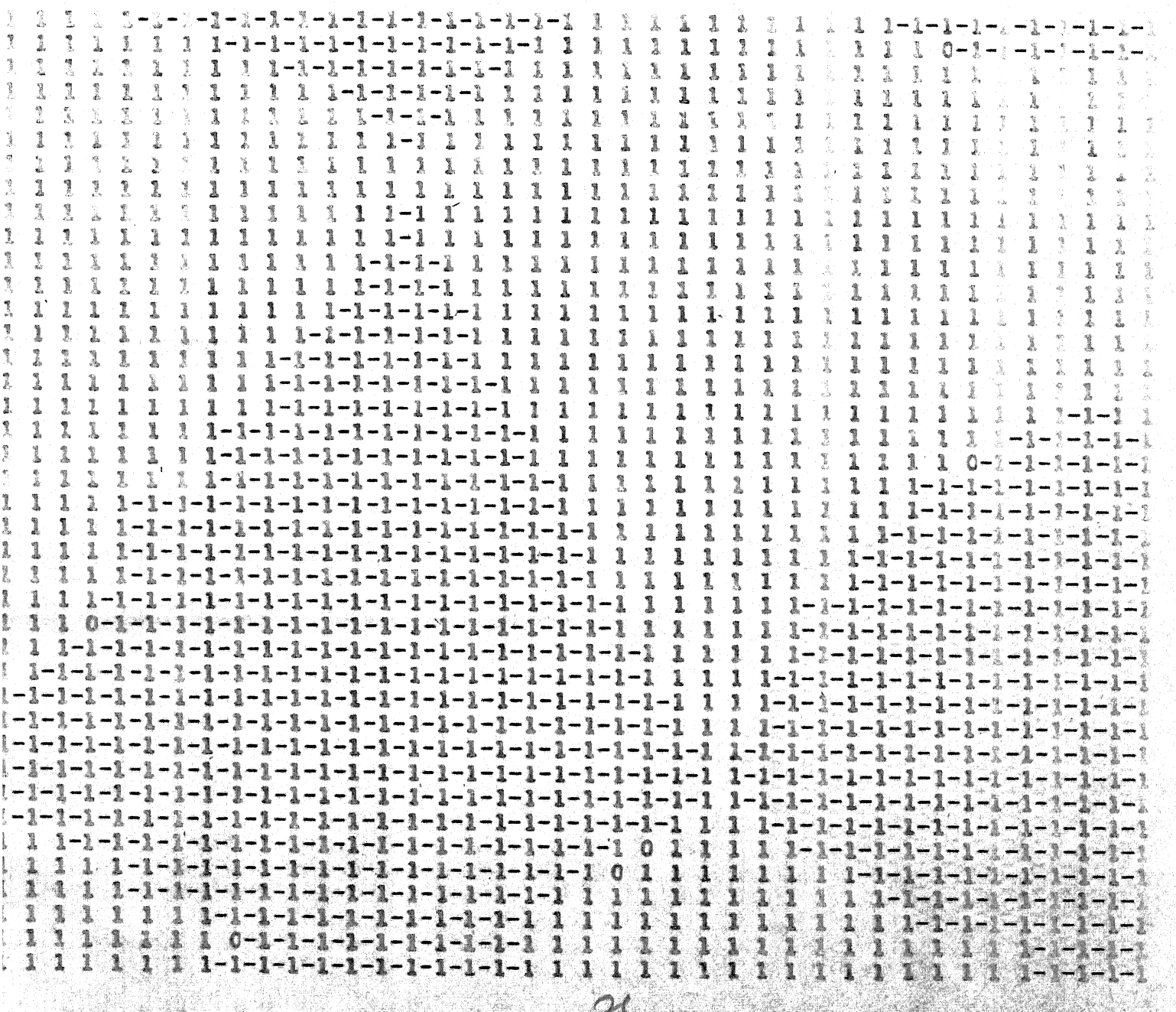


Fig. 3.18: Generalization map of trained time optimal controller with normal 01 contrast code. (Training set of Table II).

Table II

An alternative training set for training time optimal learning controller

S. No.	X axis quantum number	Y axis quantum number	Control	S. No.	X axis quantum number	Y axis quantum number	Control
1	2	4	+1	26	8	40	-1
2	3	39	+1	27	9	40	-1
3	4	39	+1	28	9	39	-1
4	5	39	+1	29	10	39	-1
5	6	38	+1	30	11	39	-1
6	7	38	+1	31	11	38	-1
7	8	38	+1	32	12	38	-1
8	9	37	+1	33	13	37	-1
9	10	37	+1	34	14	37	-1
10	11	36	+1	35	14	36	-1
11	12	35	+1	36	15	35	-1
12	13	35	+1	37	27	8	-1
13	14	34	+1	38	28	7	-1
14	26	7	+1	39	29	6	-1
15	26	6	+1	40	30	6	-1
16	27	6	+1	41	31	5	-1
17	28	5	+1	42	32	5	-1
18	29	4	+1	43	33	4	-1
19	30	4	+1	44	34	4	-1
20	31	3	+1	45	35	3	-1
21	32	3	+1	46	36	3	-1
22	34	2	+1	47	37	2	-1
23	35	1	+1	48	38	2	-1
24	36	1	+1	49	39	2	-1
25	6	40	-1	50	40	1	-1

The number of corrections needed for various codes were 577, 1300 and 177 for Adaline using normal identity, 01 contrast and ± 1 contrast codes respectively. It will be noticed that contrast type codes have a strong tendency to generalize correctly in the zone of partial knowledge and zone of ignorance. This property is very prominent in case of ±1 contrast codes (Fig. 3.19).

Learning characteristics of Adaline with relaxation algorithm:

We have mentioned earlier that the fluctuation of a priori probability of correct classification after a large number of corrections could be due to non-optimum movements of the partitioning hyperplane in fixed increment error correction method. The relaxation algorithm is a special form of error correction method in which the correction coefficient is variable between the successive errors. Thus

$$\underline{W}(k+1) = \underline{W}(k) - \eta\, \underline{S}(k), \quad \text{if } \underline{W}^T(k)\, \underline{S}(k) \leq 0$$

and

$$\underline{W}'k+1) = \underline{W}(k) \quad \text{if } \underline{W}^T(k)\, \underline{S}(k) > 0 \qquad (3.11)$$

where

$$\eta = \lambda \frac{\underline{W}^T(k).\underline{S}(k)}{||\underline{S}(k)||^2}$$

It will be noticed that will be large or small depending on the magnitude of $\underline{W}^T(k).\underline{S}(k)$. This helps the

learning process to converge faster. We shall briefly examine the convergence characteristics of this algorithm for different codes. Table III shows the training samples of Table I, but in a different sequence. It was found during the experiments that the sequence of training also affects the convergence rate. Keeping this in mind another training sequence was also tried in which the training samples were presented in a random manner, i.e. a random number between 1 to 50 was generated and accordingly the corresponding training sample was presented to the Adaline. Graphs were plotted to investigate the nature of convergence with respect to λ . As shown in Fig. 3.20, 3.21 and 3.22, the convergence is faster in case of random presentation of training samples as compared to sequential presentation. The convergence is extremely fast in case of ± 1 contrast code, and further for training set of Table I, Adaline using 01 contrast code exhibits discontinuities at $\lambda = 1.0$ and 1.5, nonconvergence being obtained even after 30 minutes of computer run. It will be noticed that in Table III, the training samples of the two classes are mixed up as compared to training samples of Table I. This suggests that for faster convergence of training, presentation of training samples should be random. Generalization characteristics of controller trained with this algorithm were very similar to that of fixed increment error correction learning algorithm.

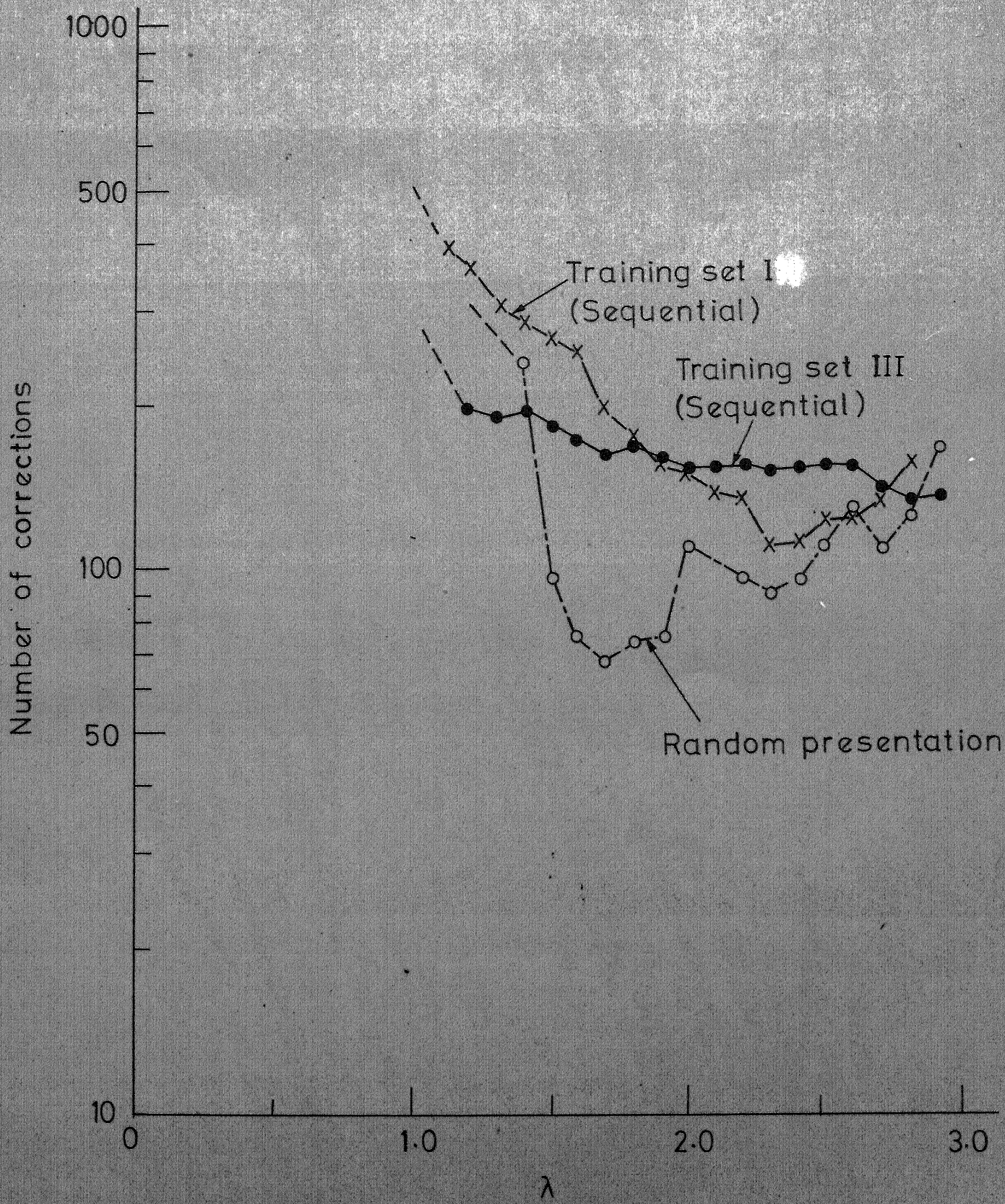


FIG. 3.20 CONVERGENCE CHARACTERISTICS OF LEARNING CONTROLLER WITH RELAXATION ALGORITHM IDENTITY CODE

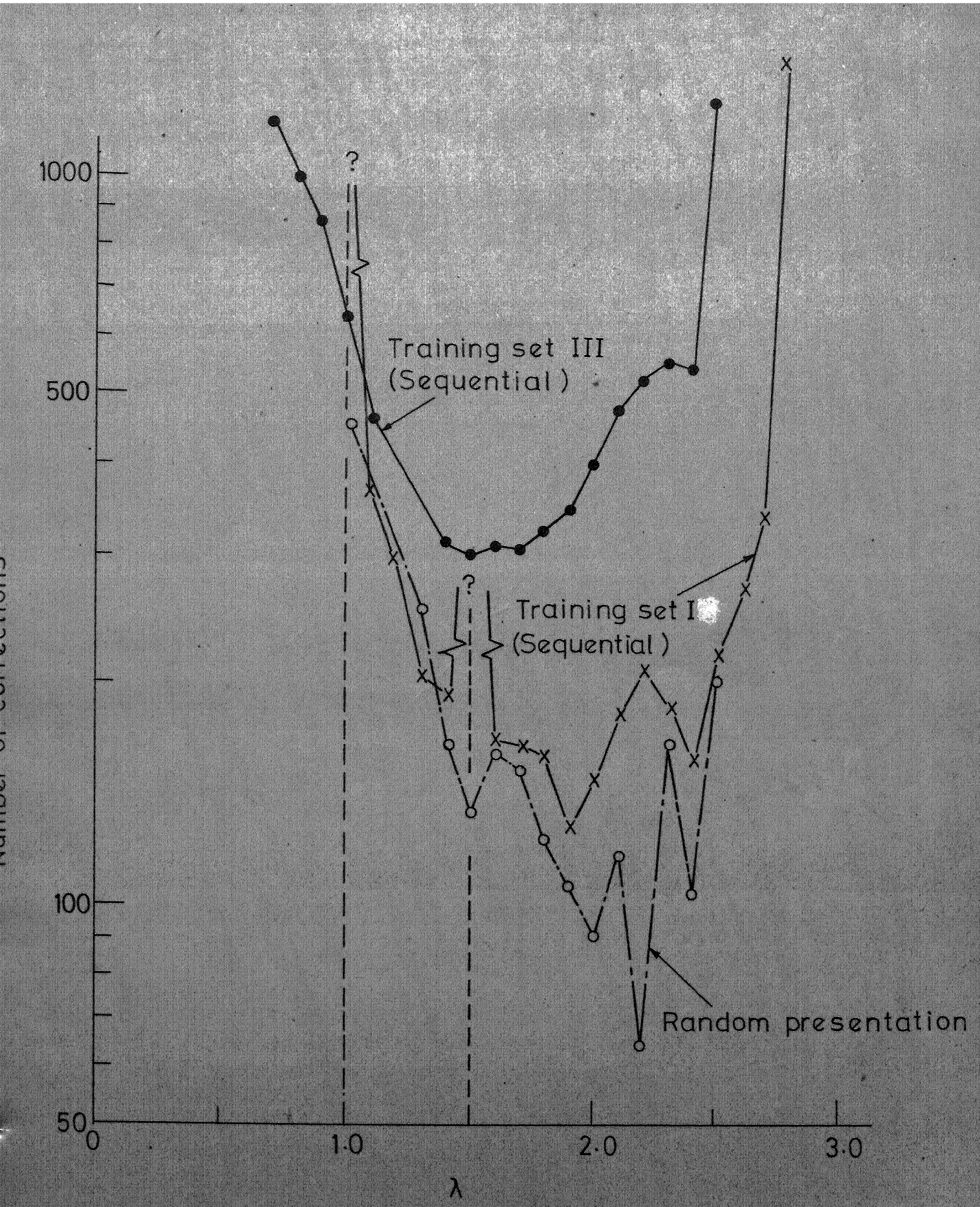


FIG. 3·21 CONVERGENCE CHARACTERISTICS OF LEARNING CONTROLLER WITH RELAXATION ALGORITHM. 01 CONTRAST CODE.

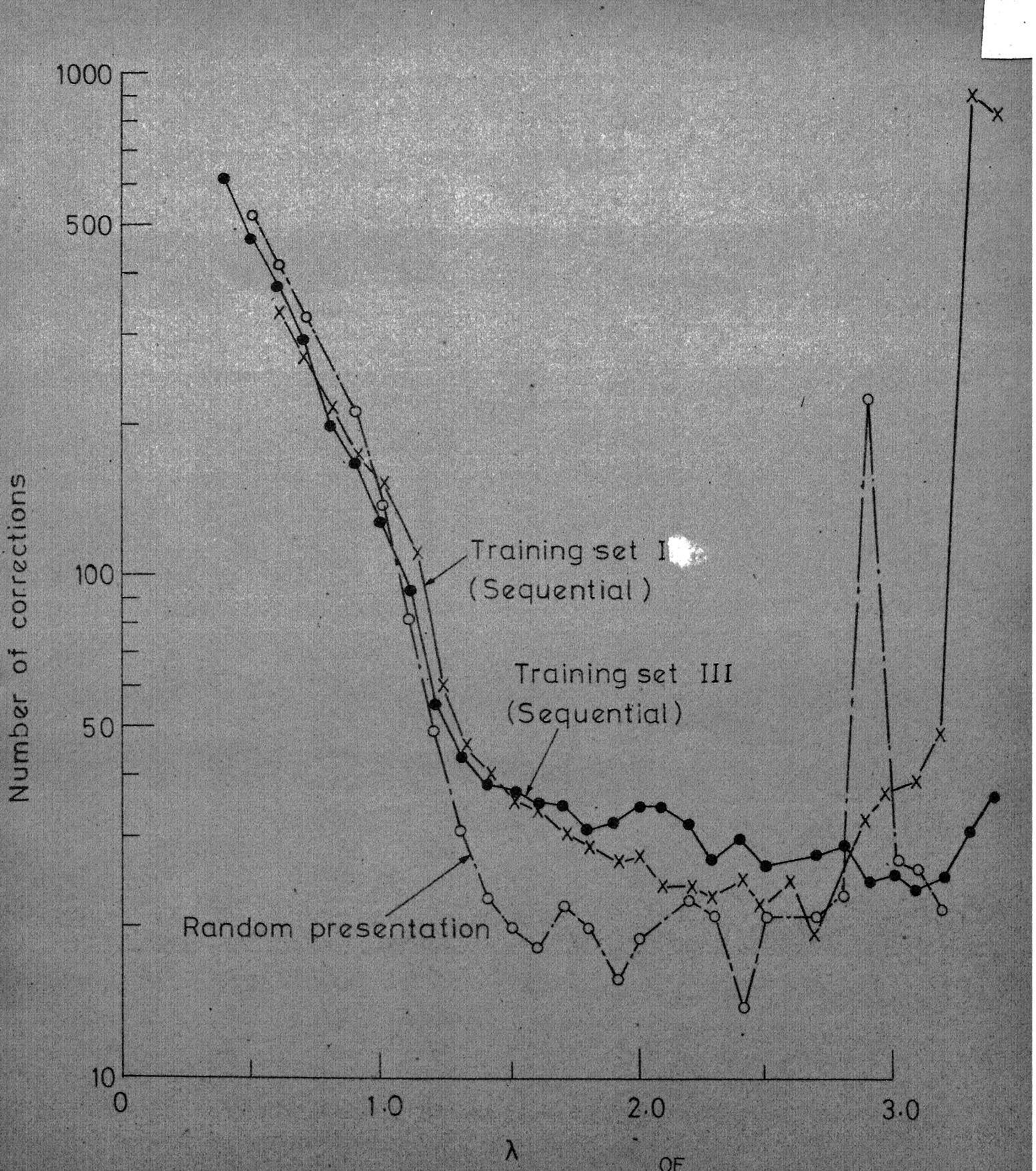


FIG. 3.22 CONVERGENCE CHARACTERISTICS OF LEARNING CONTROLLER WITH RELAXATION ALGORITHM ±1 CONTRAST CODE.

Table III

A training set for time optimal learning controller

S. No.	X axis quantum number	Y axis quantum number	Control	S. No.	X axis quantum number	Y axis quantum number	Control
1	26	12	+1	26	14	24	+1
2	26	10	+1	27	18	24	+1
3	31	9	+1	28	14	26	+1
4	15	29	-1	29	17	26	+1
5	22	30	-1	30	14	22	+1
6	17	28	-1	31	14	30	-1
7	10	32	-1	32	27	13	-1
8	24	28	-1	33	18	22	+1
9	19	26	-1	34	24	13	+1
10	24	26	-1	35	20	14	+1
11	20	25	-1	36	16	21	+1
12	26	23	-1	37	20	12	+1
13	26	20	-1	38	22	12	+1
14	22	15	+1	39	14	11	+1
15	22	20	-1	40	25	14	-1
16	28	18	-1	41	29	10	+1
17	22	18	-1	42	29	12	-1
18	30	10	-1	43	31	11	-1
19	30	16	-1	44	21	26	-1
20	23	16	-1	45	20	21	+1
21	30	24	-1	46	16	19	+1
22	30	19	-1	47	20	19	+1
23	10	30	+1	48	16	17	+1
24	12	29	+1	49	20	17	+1
25	14	28	+1	50	12	31	-1

3.5 Learning with Three Levels of Control: Fuel Optimal Learning Control Systems:

As pointed out in Chapter 1, the fuel optimal controllers are of bang-off-bang nature; i.e. there are three stable states of the control. Since an Adaline can generate only two states of control, it is therefore necessary that the control be decoded by using a two bit word with a minimum of two Adalines for a fuel optimal learning controller. Fig. 3.23 shows the setup to realize the fuel optimal learning controller.

Fig. 3.24 shows an arbitrary fuel optimal switching dichotomy selected for training the controller. The states were quantized in 40 quantum levels each and training samples were chosen from the quantized state space. Table IV shows the training set.

Mendel [29] has described a method of training for the fuel optimal learning controller shown in Fig. 3.22. The method is as follows.

Let u_I and u_{II} be the responses of the Adaline I and II respectively. Let us assume that we are having a training set of patterns with known classification, i.e. R = +1, 0, or -1. We wish that a two bit word from the Adalines be presented to the decoder as shown in Fig. 3.23. Thus if we assume that the output of an Adaline is either

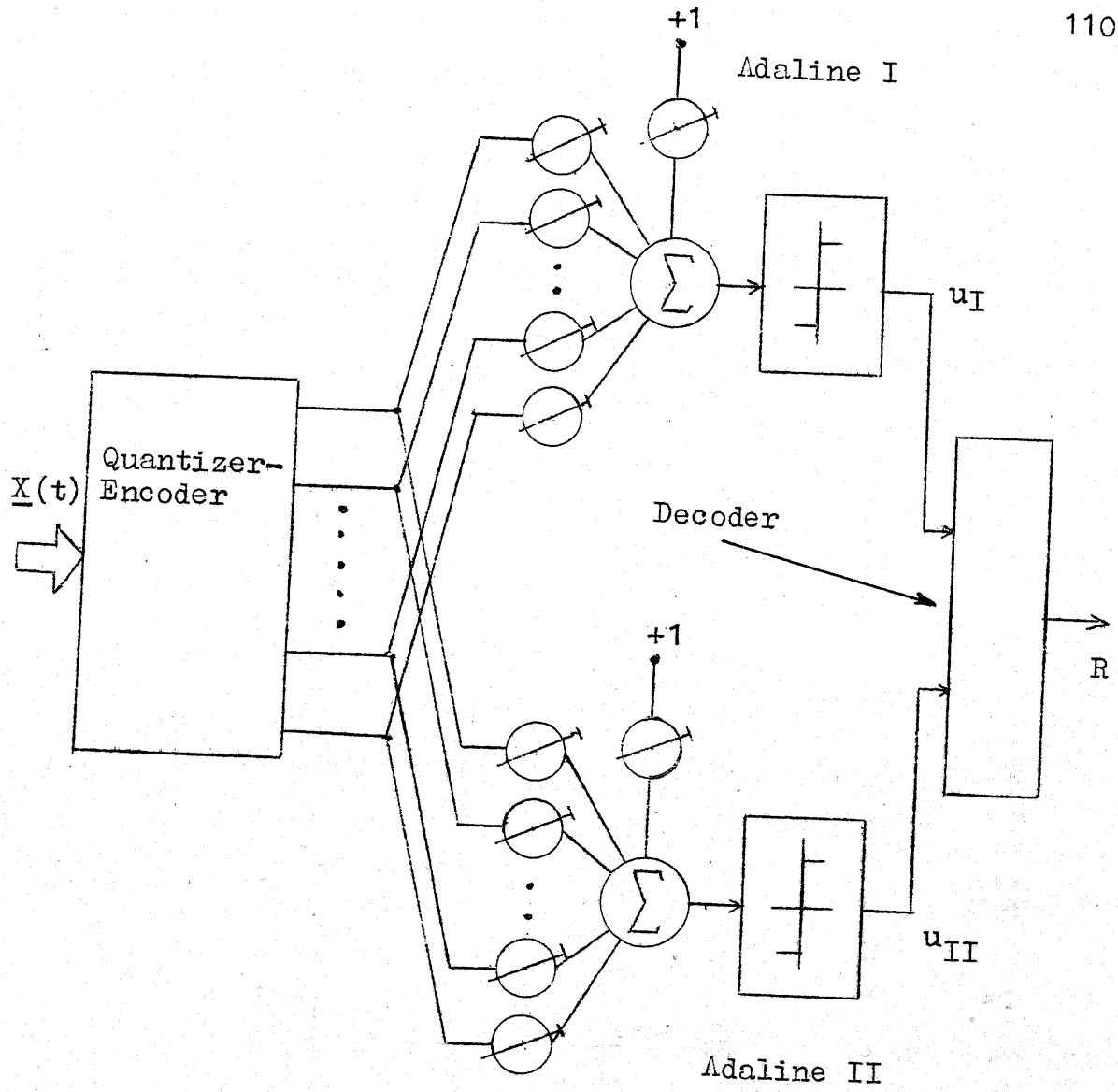


Fig. 3.23 A network of Adalines to realize a fuel optimal control law.

Table IV

A training set for training the fuel optimal controller

S. No.	X axis quantum number	Y axis quantum number	Control	S. No.	X axis quantum number	Y axis quantum number	Control
1	2	32	+1	44	22	17	0
2	2	33	0	45	22	18	0
3	4	33	0	46	23	15	0
4	6	31	+1	47	23	16	0
5	7	32	0	48	24	13	0
6	8	30	+1	49	25	15	0
7	10	31	0	50	26	14	0
8	11	29	+1	51	26	11	0
9	12	30	0	52	27	10	0
10	14	29	0	53	28	9	0
11	14	27	+1	54	30	8	0
12	15	26	+1	55	31	7	0
13	16	27	0	56	33	6	0
14	16	25	+1	57	35	5	0
15	17	24	+1	58	37	4	0
16	18	25	0	59	36	3	+1
17	18	22	+1	60	34	4	+1
18	19	23	0	61	32	5	+1
19	19	24	-0	62	30	6	+1
20	20	25	-1	63	28	7	+1
21	20	26	-1	64	27	8	+1
22	18	27	-0	65	24	11	+1
23	19	28	-1	66	23	12	+1
24	17	29	-0	67	22	13	+1
25	18	30	-1	68	39	9	0
26	16	30	-0	69	36	10	0
27	17	31	-1	70	35	10	0
28	15	31	-0	71	32	11	0
29	16	33	-1	72	39	12	0
30	13	33	0	73	28	13	0
31	12	34	-0	74	26	14	-0
32	13	36	-1	75	39	11	-1
33	11	35	0	76	34	12	-1
34	10	36	-0	77	31	13	-1
35	15	34	-1	78	29	14	-1
36	11	37	-1	79	28	15	-1
37	10	38	-1	80	26	16	-1
38	7	38	-0	81	25	17	-1
39	8	39	-1	82	24	18	-1
40	5	39	0	83	23	19	-1
41	20	17	+1	84	32	13	-1
42	21	19	0	85	37	10	0
43	21	15	+1				

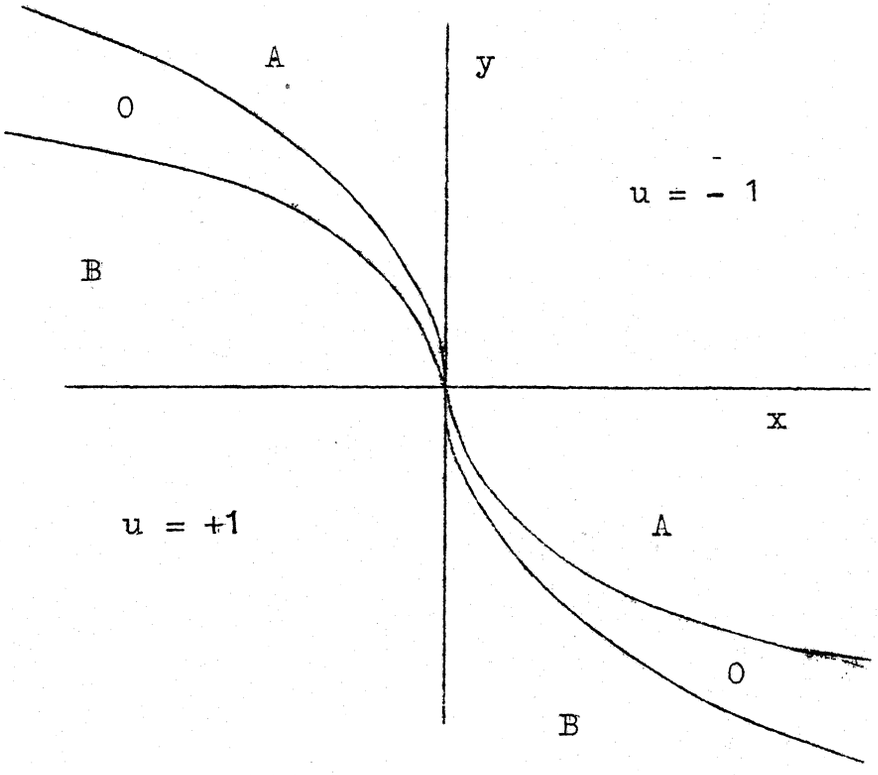


Fig. 3.24 Arbitrary fuel optimal switching dichotomy selected for training the Adaline controllers.

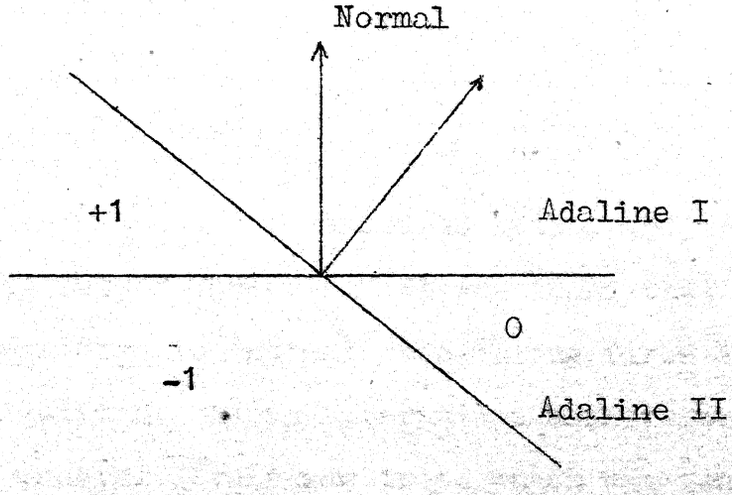


Fig. 3.25 Mapping of control levels achieved by Table V.

+1 or -1, then there are $2^2 = 4$ distinct words that can be presented as input to the decorder. If we can assign a distinct word to each control level then the response of the Adalines can be mapped onto one of the control levels. Mendel [29] has suggested that if the control levels are coded as shown below in Table V, the network of Fig. 3.23 can be trained on any linearly separable set.

Table V

Coding scheme for control levels as suggested by Mendel

Response of Adaline I	Response of Adaline II	Control level
+1	-1	+1
-1	+1	0
-1	-1	-1

The network of Adalines shown in Fig. 3.23 was simulated using the above described method and trained on the training set of Table IV. It was found that the method failed to converge for normal codes using fixed increment training algorithm. As an alternative a different coding scheme for control levels was tried which was found to converge extremely fast. This coding scheme is as shown in Table VI.

Table VI

Coding scheme for control levels

Response of Adaline I	Response of Adaline II	Control level
+1	+1	+1
-1	+1	0
-1	-1	-1

Another coding scheme which is also found to be convergent is shown in Table VII.

Table VII

Alternative coding scheme for control levels

Response of Adaline I	Response of Adaline II	Control level
+1	-1	+1
-1	-1	0
-1	+1	-1

It will be noticed from Fig. 3.24 that the control levels +1 and -1 are separated by the control level 0. Thus if training samples are chosen close to the switching surface then, the stimuli chosen near the surface A will belong to either the -1 or 0 control level and will have

closely similar patterns. Similarly the samples chosen near the surface B will either belong to 0 or +1 control levels and will have similarity in patterns. From the mapping as shown in Table V the separating hyperplanes realized by the two Adalines will have a form as shown in Fig. 3.25. It is clear from Fig. 3.25 that the stimuli of the +1 and -1 class near the partitioning surface due to Adaline I should be similar, which is contrary to Fig. 3.24. This will lead to the nonconvergence of the training since there will not exist partitioning hyperplanes to realize the training samples.

The fuel optimal learning controller of Fig. 3.23 was simulated using fixed increment error correction algorithm Training was accomplished on the training set of Table IV for different normal codes. Fig. 3.26, 3.27 and 3.28 show the generalization characteristics for various codes. It will be noted that the generalization due to identity code is poor as compared to the contrast codes. In Fig. 3.27 and 3.28, poor definition of dichotomies near origin occurs due to lack of training samples near origin.

To investigate the generalization property of codes, random codes were also tried. It was noticed that with contrast type codes, the controller usually failed to converge. Fig. 3.29 demonstrates a typical ± 1 contrast code matrix for which the controller failed to converge.

Fig. 3.26: Generalization map of the trained fuel optimal controller with normal identity code. (Training set of Table IV). Numbers other than 0, -1 or +1 indicate incorrect response combinations of the Adalines.

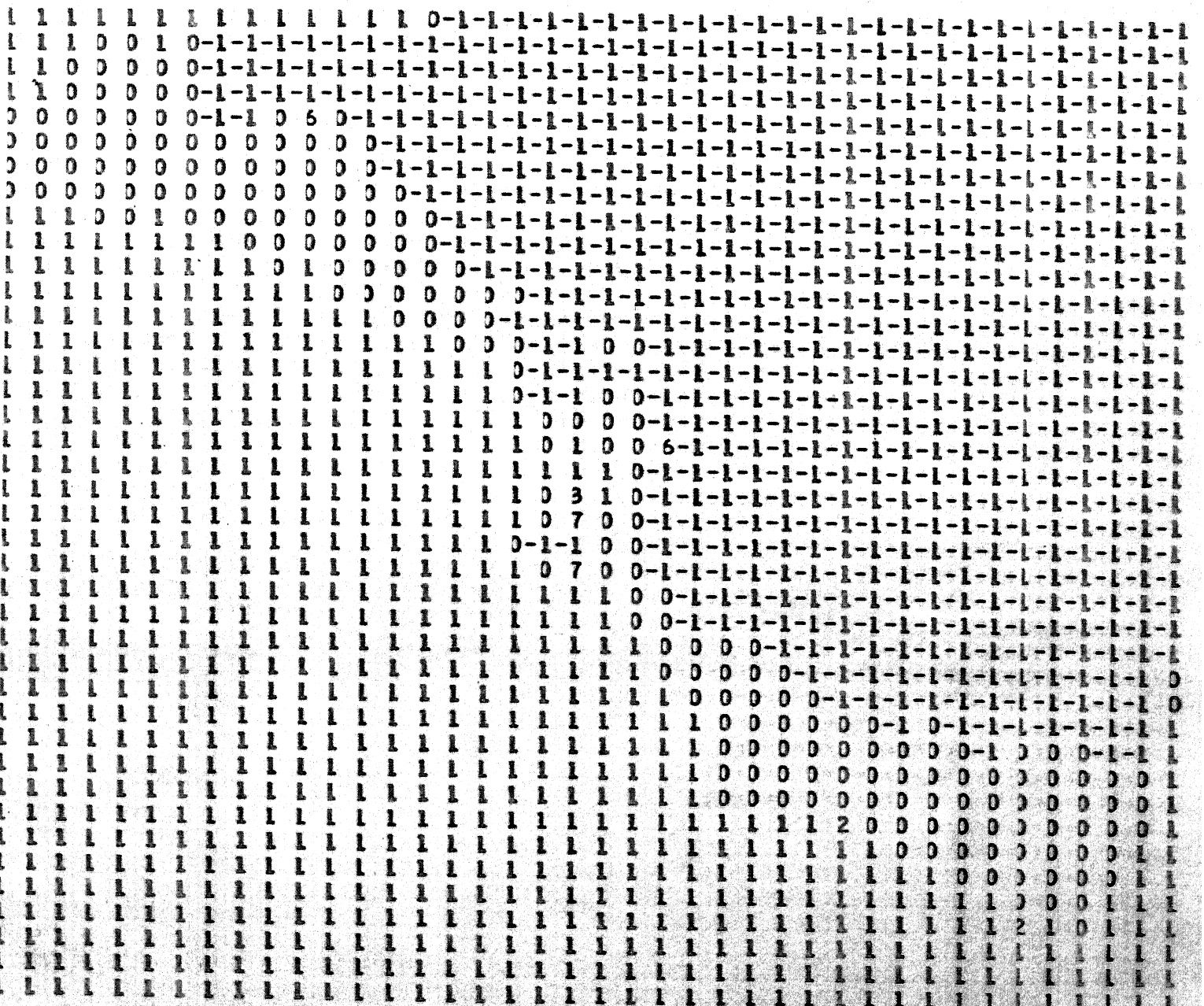


Fig. 3.27: Generalization map of trained fuel optimal controller with normal 01 contrast code. (Training set of Table IV). Numbers other than 0, -1 or +1 indicate incorrect response combinations of the Adalines.

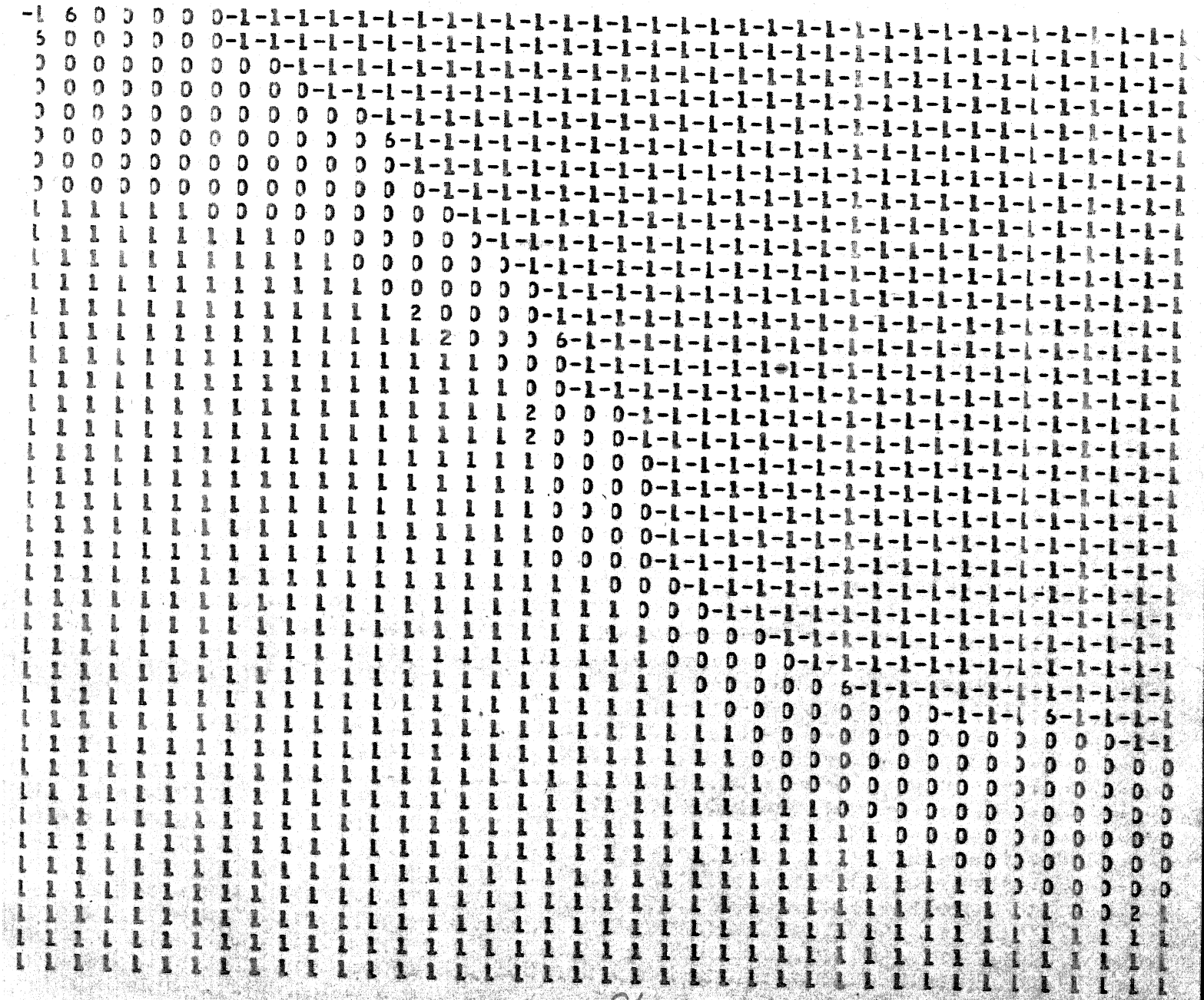


Fig. 3.28: Generalization map of the trained fuel optimal controller with normal ± contrast code. (Training set of Table IV). Numbers other than 0, -1 and +1 indicate incorrect response combinations of the Adalines.

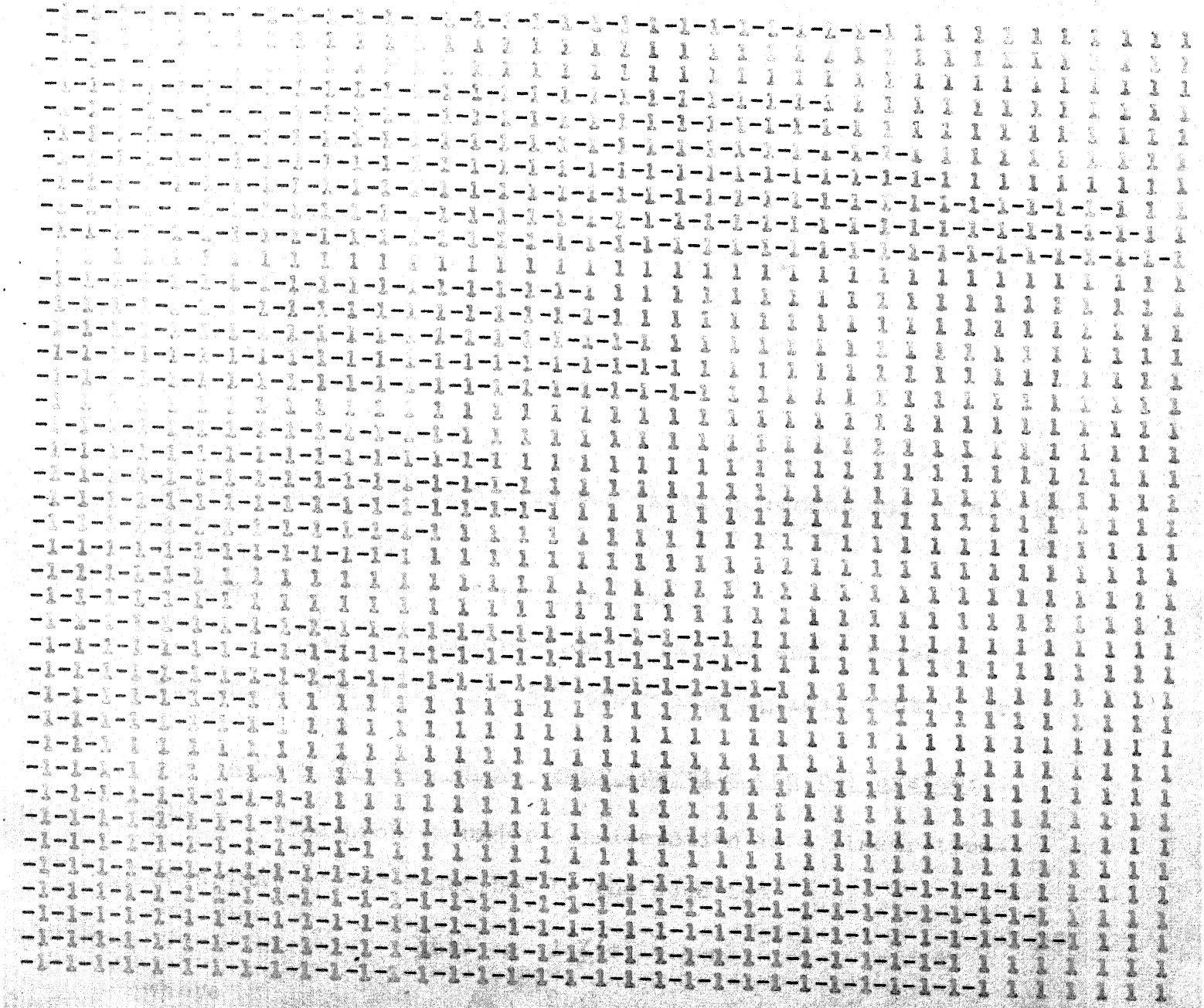


Fig. 3.29: A typical random code matrix for which the fuel optimal learning controller fails to attain a learned state.

Fig. 3.30 and 3.31 demonstrate the contrast random code matrices for which the controller converges to a learned state. The corresponding generalization maps are shown in Fig. 3.32 and 3.33.

From the previous discussion it is apparent that a "good" generalization is guaranteed if the normal contrast type of code is used to code the quantum levels. A faster convergence of the learning process is however achieved with ± 1 contrast type of code. Improvement in convergence is obtained by the use of relaxation algorithm. It was also noticed that there is no significant difference in generalization characteristics with the fixed increment and relaxation algorithms. Improvement in convergence is achieved with random presentation of training samples.

With this information in hand we shall investigate the characteristics of a 3^{rd} order time optimal controller.

3.6 A Time Optimal Third Order Learning Control System:

The problem under consideration is a linear time-invariant system described by the equation

$$\dot{\underline{X}}(t) = A\,\underline{X}(t) + \underline{b}\,u$$

where

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

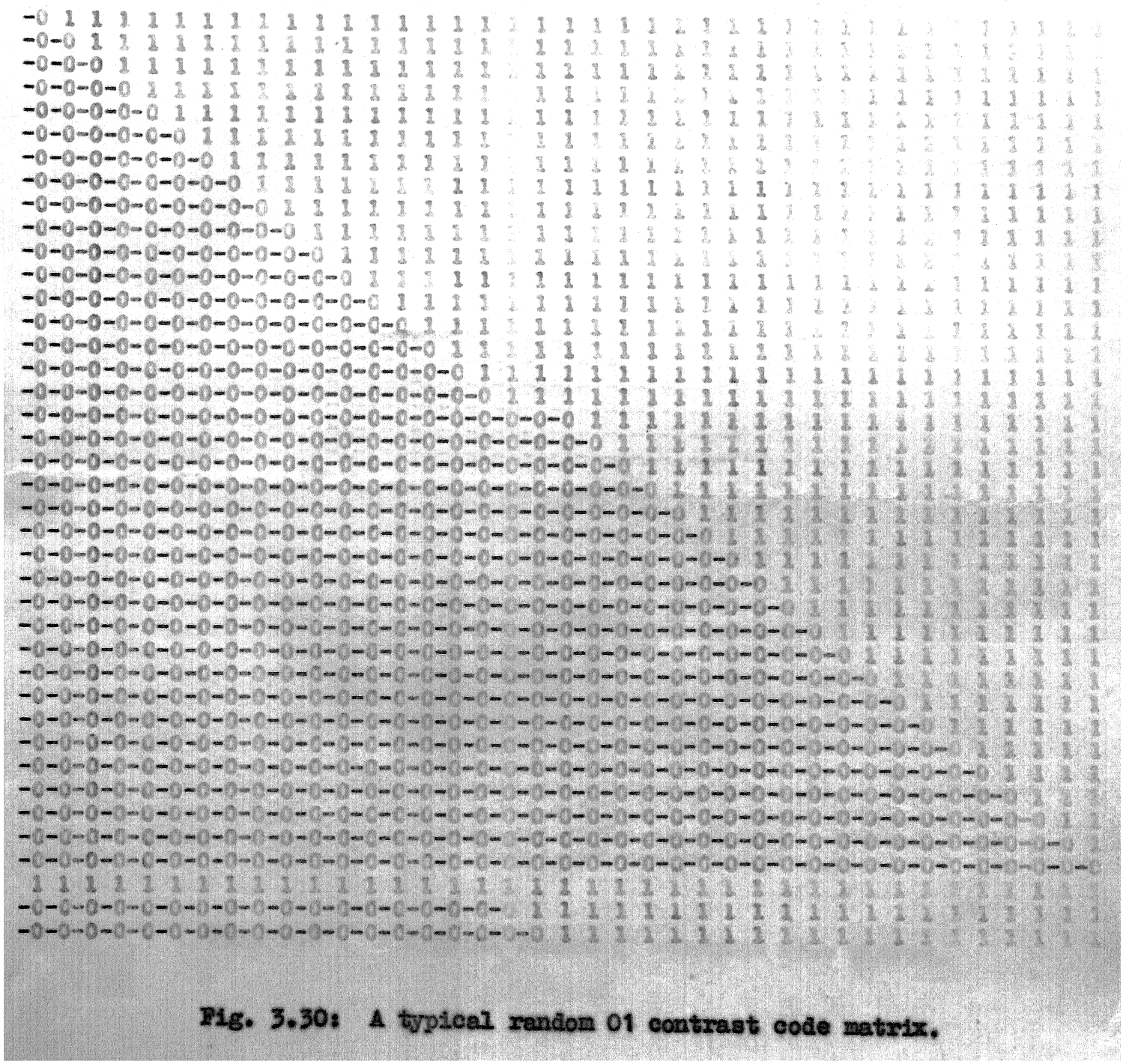


Fig. 3.30: A typical random 01 contrast code matrix.

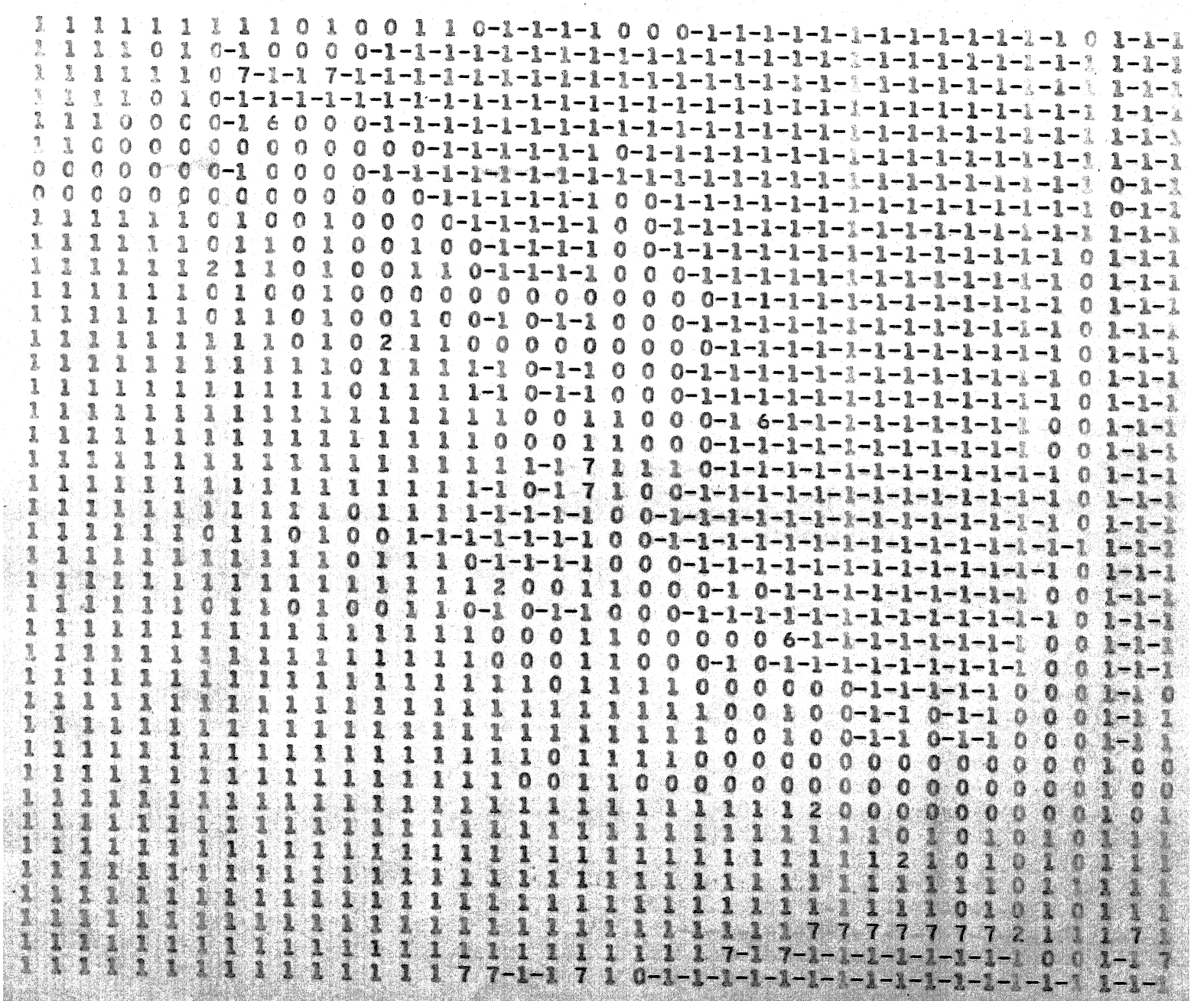


Fig. 3.32: Generalization map of the trained fuel optimal controller with random code matrix of Fig. 3.30 and training set of Table IV. Numbers other than 0, -1 and +1 indicate incorrect response combinations of the Adalines.

Fig. 3.33: Generalization map of the trained fuel optimal controller with random code matrix of Fig. 3.31 and training set of Table IV. Numbers other than 0, -1 or +1 indicate incorrect response combinations of the Adalines.

and

$$\underline{b} = [0 \quad 1 \quad 1]^T$$

This system is a simplified re-entry vehicle model described by Mendel [30].

Here the basic problem is to drive the system from an initial state $\underline{X}(0)$ to the origin in minimum time.

The region of interest in state space was chosen as $|x_1| \leq 10$, $|x_2| \leq 6$, and $|x_3| \leq 25$. All the three states were quantized into 40 quantum levels, thus forming 64000 quantum control situations in state space. Encoding of quantum levels was done with normal codes.

As previously stated, the open loop, time optimal control is the control $u^*(t)$ which satisfies (1.8) and (1.10) such that terminal time t_f is a minimum. Neustadt 3 has developed an iterative method to compute the control $u^*(t)$ to generate the optimal trajectories in state space. The details of this method can be found in Mendel [30].

A parallelopiped of sides $X_1 = 5$, $X_2 = 3$ and $X_3 = 12.5$ was selected centered around the origin. The vertices of this parallelopiped were chosen as initial conditions for generating the optimal trajectories.

The training set includes those cubes (patterns) through which the optimal trajectories pass. A computer program generated the optimal trajectories and the various

control situations were identified through which the optimal trajectories passed. The computer program rejected redundant control situations (i.e. control situations already encountered).

The Adaline controller using fixed increment error correction method was simulated and was trained on the training set as obtained above. The training set included 560 training samples forming 0.875% of the total control situations. The training was found to be convergent for ± 1 contrast code within 1079 iterations although convergence for identity and 01 contrast codes was not obtained for iterations below 2000. This suggests that the surface is projectable. Training was terminated arbitrarily after 1000 iterations for all cases of codes.

Fig. 3.34 to 3.42 are the typical time response characteristics for initial conditions not included in the training set, using different codes. It will be noticed that the departure from optimality is small. The unreliable nature of Adaline using identity code is very clearly indicated in Fig. 3.34, 3.37 and 3.40. It will be noticed too that for identity code the Adaline changes control very rapidly. An examination of u_T^* (optimal) and u_L (Adaline control) shows that they are same until the state variables come within about two quantum control situations of the origin, thereafter u_L becomes very erratic. This results from poor definition

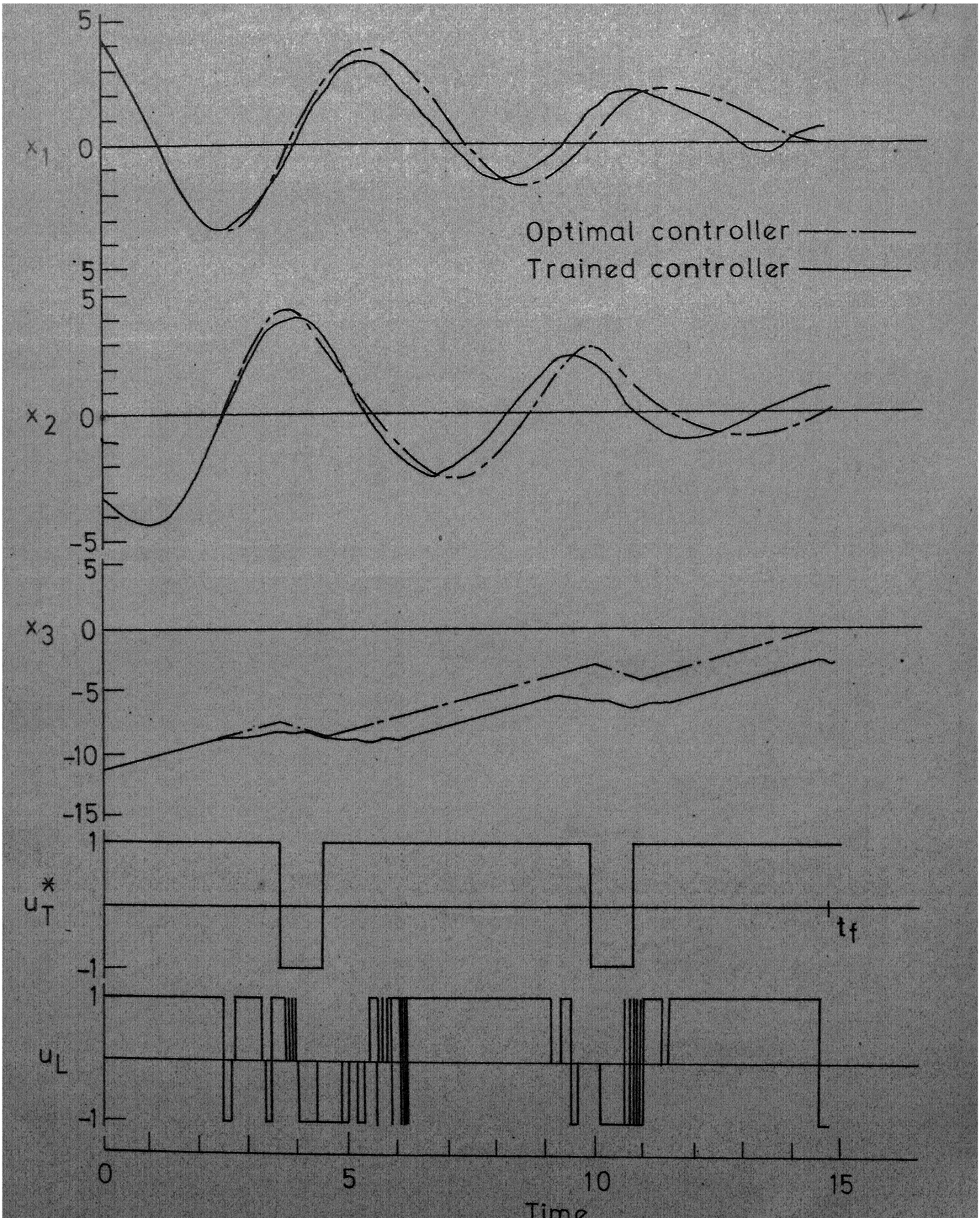


FIG. 3.34 TIME RESPONSE OF TRAINED CONTROLLER WITH IDENTITY CODE. $\underline{X}(0) = [4.1, -3.2, -11.8]^T$

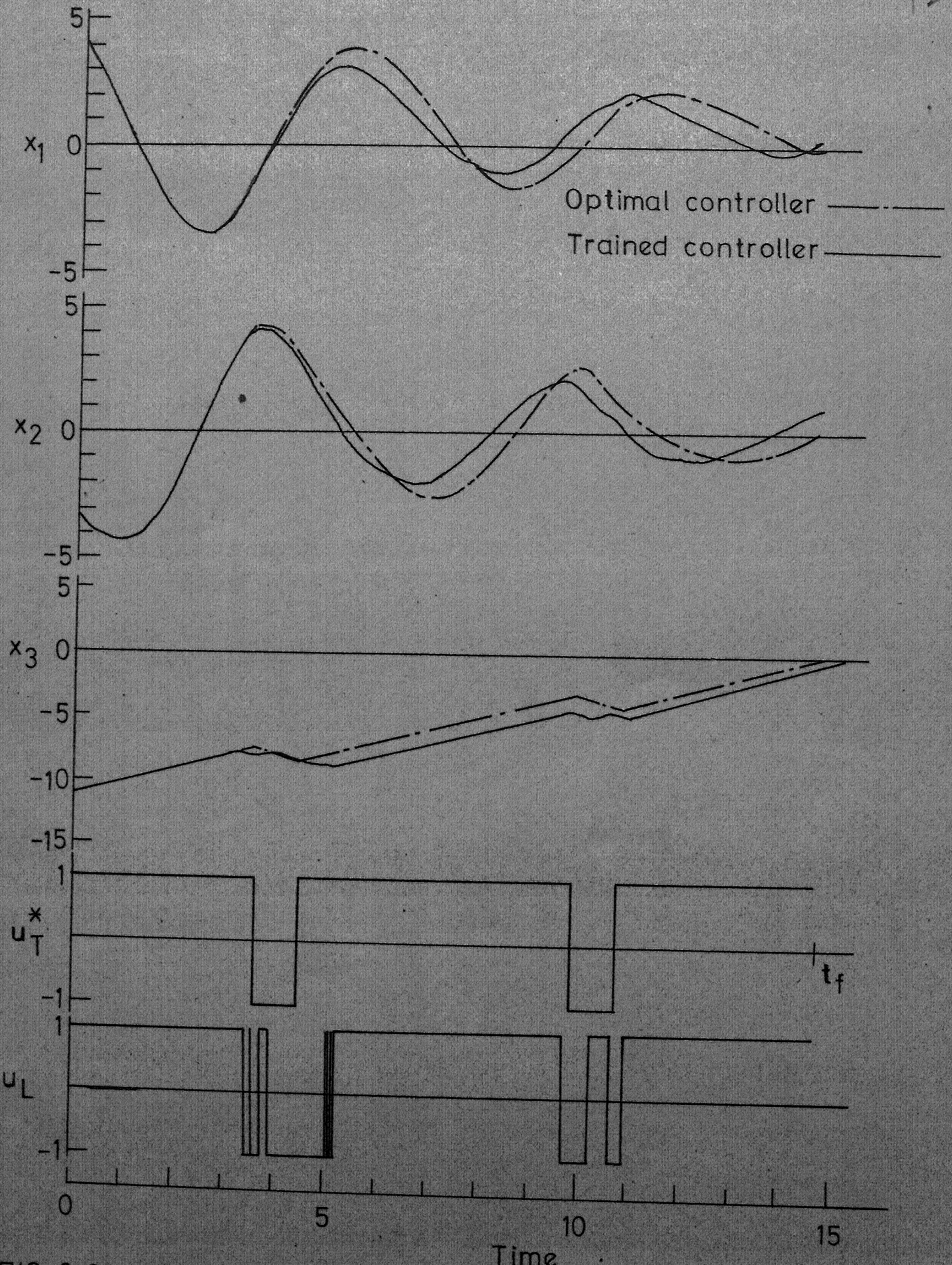


FIG. 3.35 TIME RESPONSE OF TRAINED CONTROLLER WITH ±1 CONTRAST CODE. $\underline{X}(0) = [4.1, -3.2, -11.0]^T$

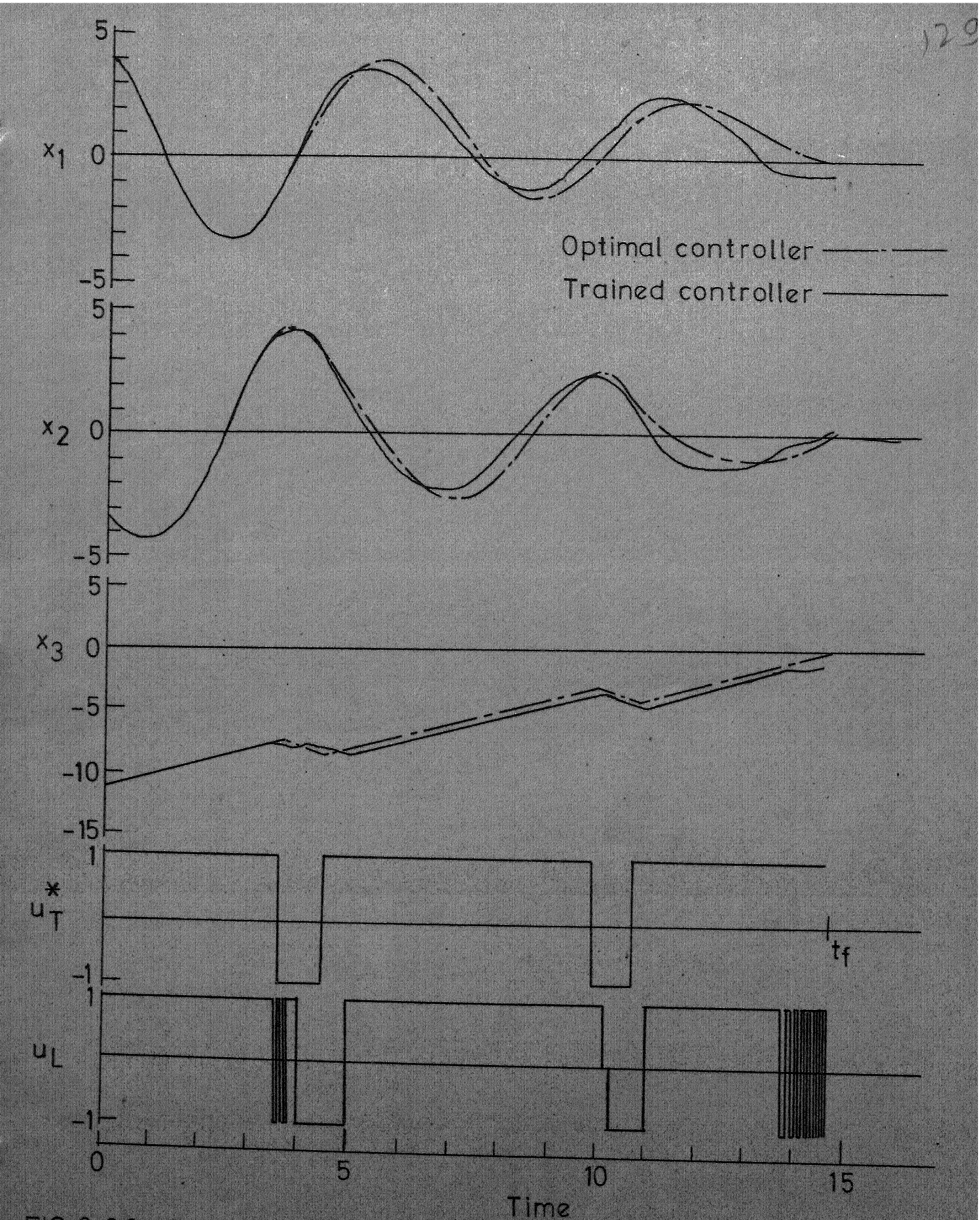


FIG. 3·36 TIME RESPONSE OF TRAINED CONTROLLER WITH 01 CONTRAST CODE. $\underline{X}(0) = [4{\cdot}1, -3{\cdot}2, -11{\cdot}0]^T$

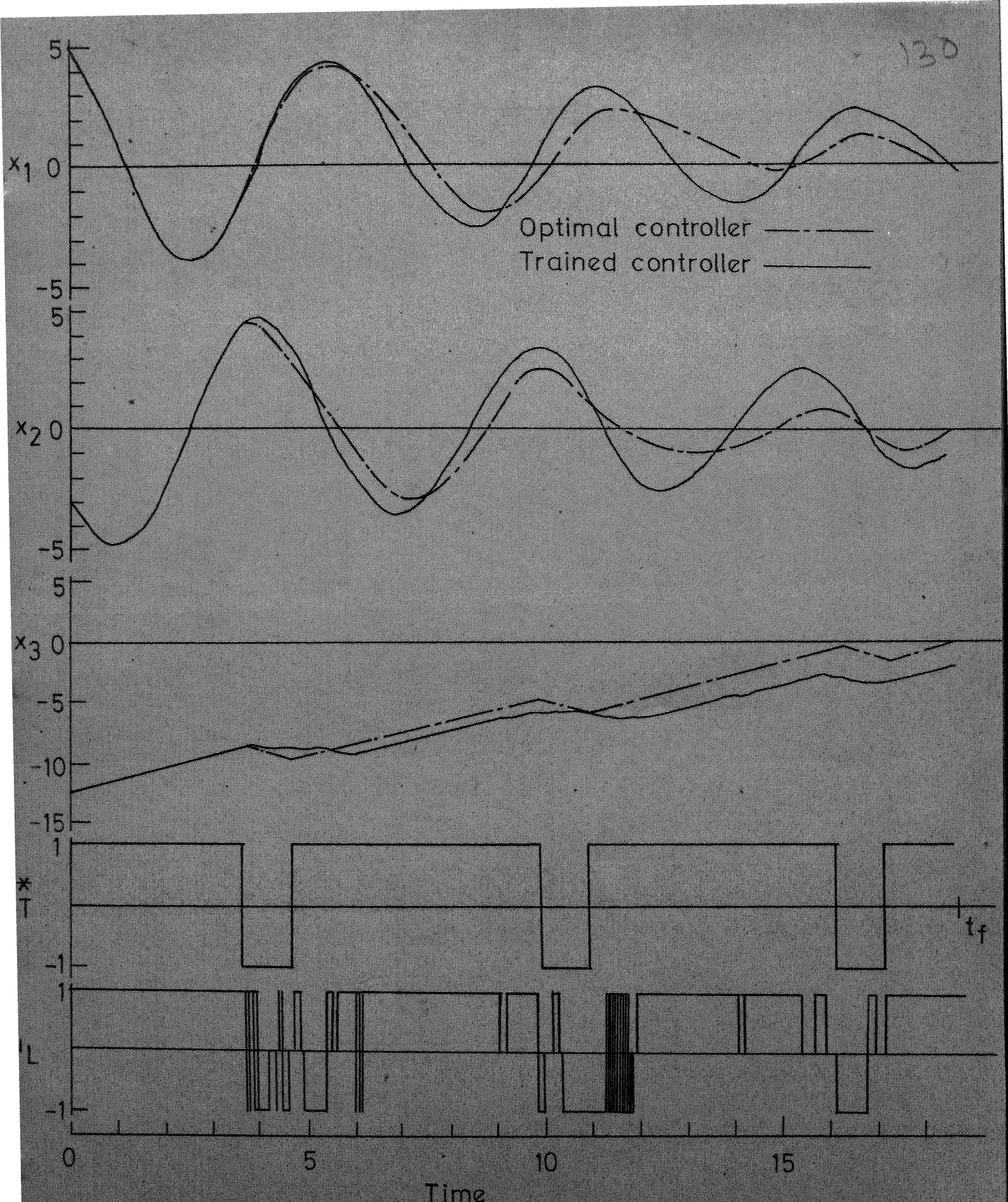
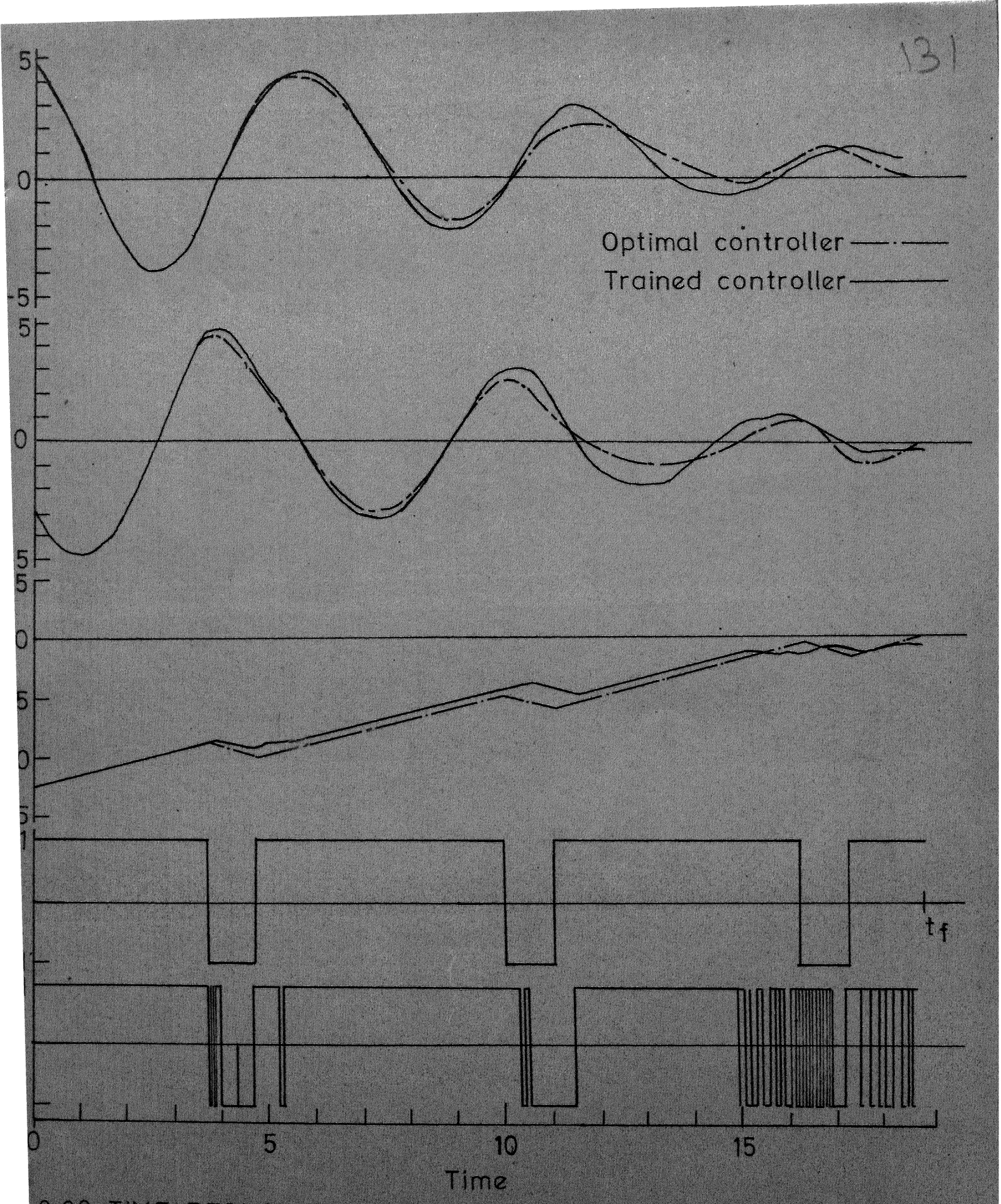
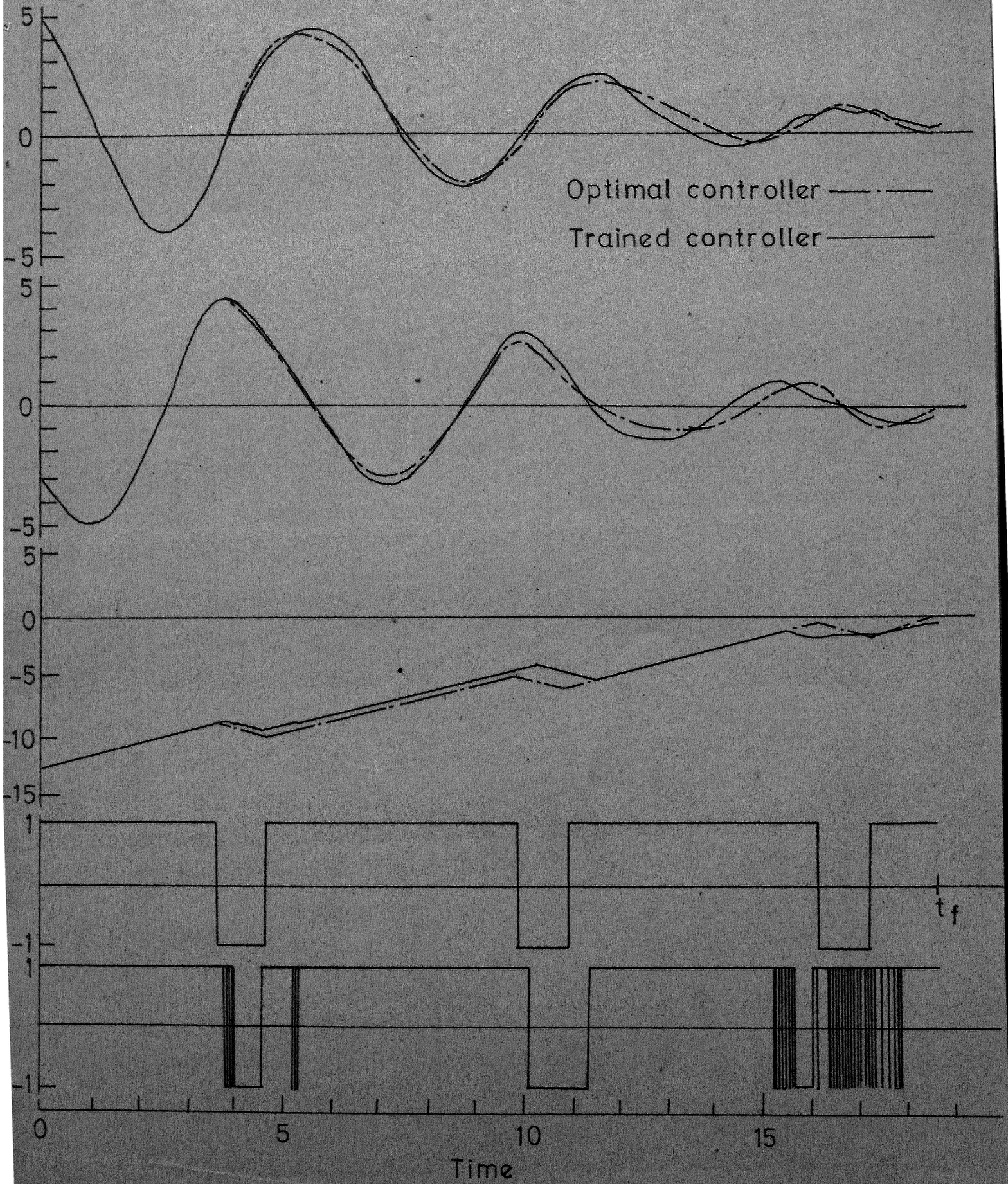


FIG. 3·37 TIME RESPONSE OF TRAINED CONTROLLER WITH IDENTITY CODE. $\underline{X}(0) = [5{\cdot}0, -3{\cdot}0, -12{\cdot}5]^T$



3·38 TIME RESPONSE OF TRAINED CONTROLLER WITH 01 CONTRAST CODE. $\underline{X}(0) = \left[5{\cdot}0, -3{\cdot}0, -12{\cdot}5\right]^T$



3.39 TIME RESPONSE OF TRAINED CONTROLLER WITH ±1 CONTRAST CODE $\underline{X}(0) = [5.0, -3.0, -12.5]^T$

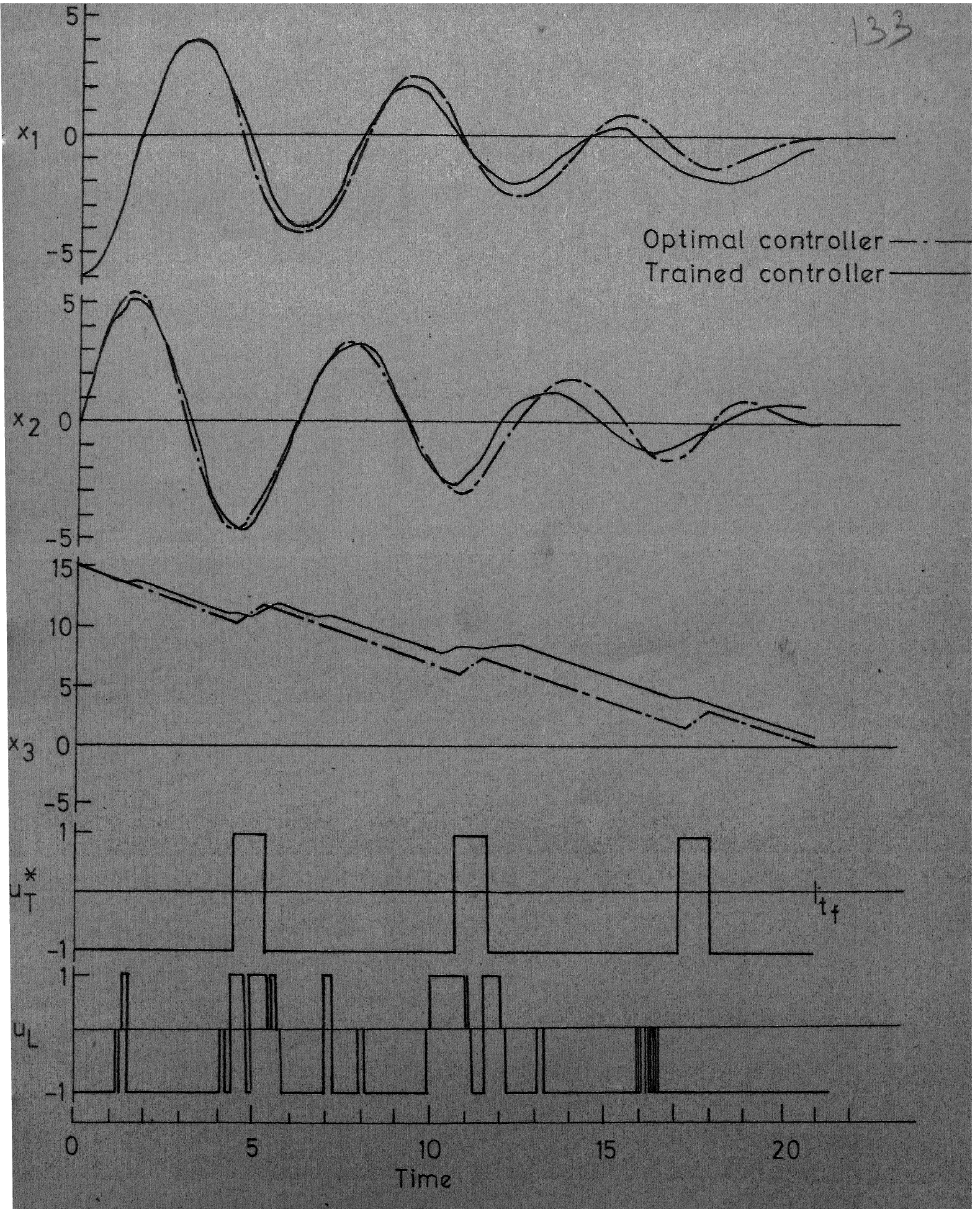


FIG. 3.40 TIME RESPONSE OF TRAINED CONTROLLER WITH IDENTITY CODE. $\underline{X}(0) = [-6.0, 0.0, 15.0]^T$

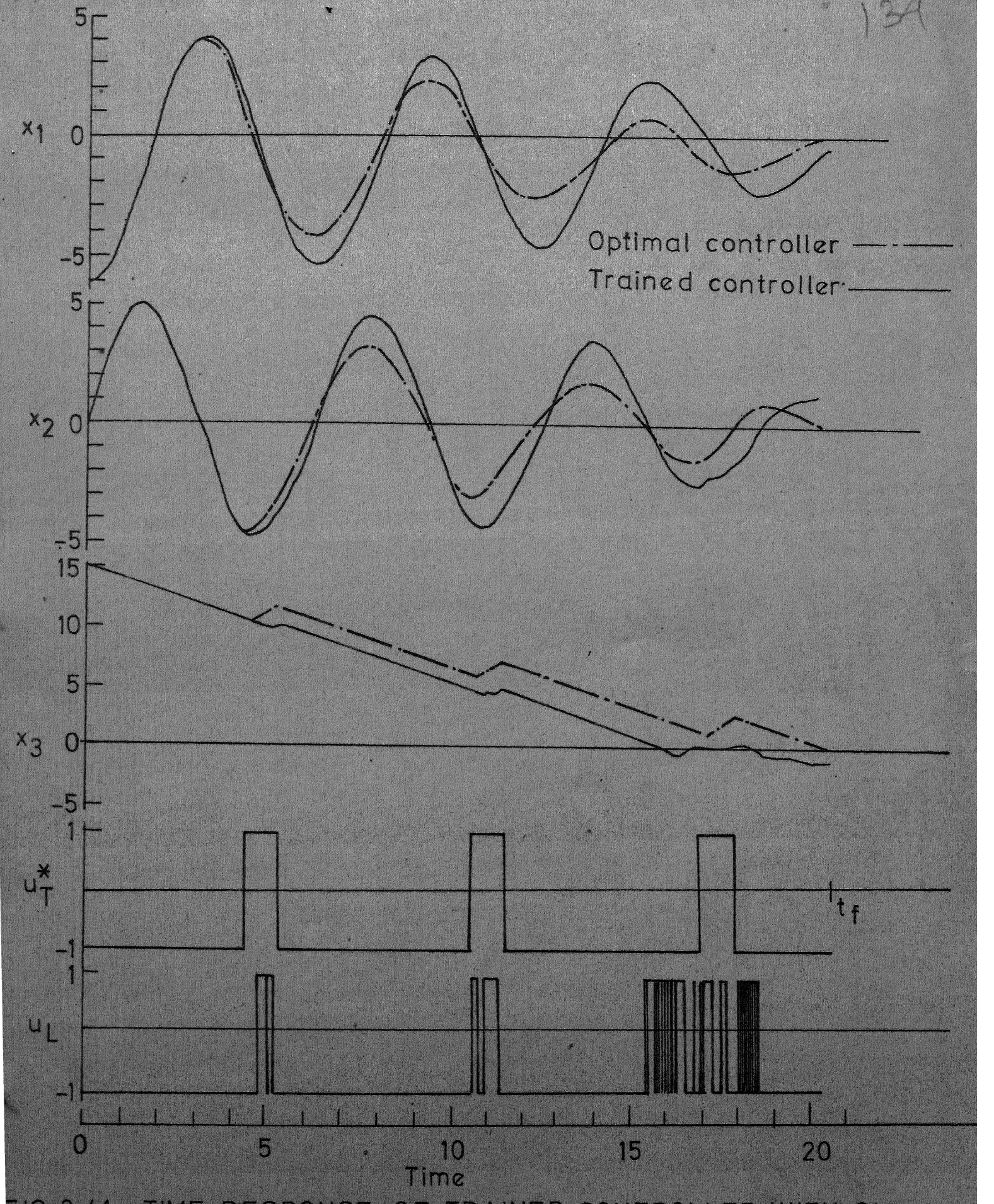


FIG. 3-41 TIME RESPONSE OF TRAINED CONTROLLER WITH 01 CONTRAST CODE. $\underline{X}(0) = [-6.0, 0.0, 15.0]^T$

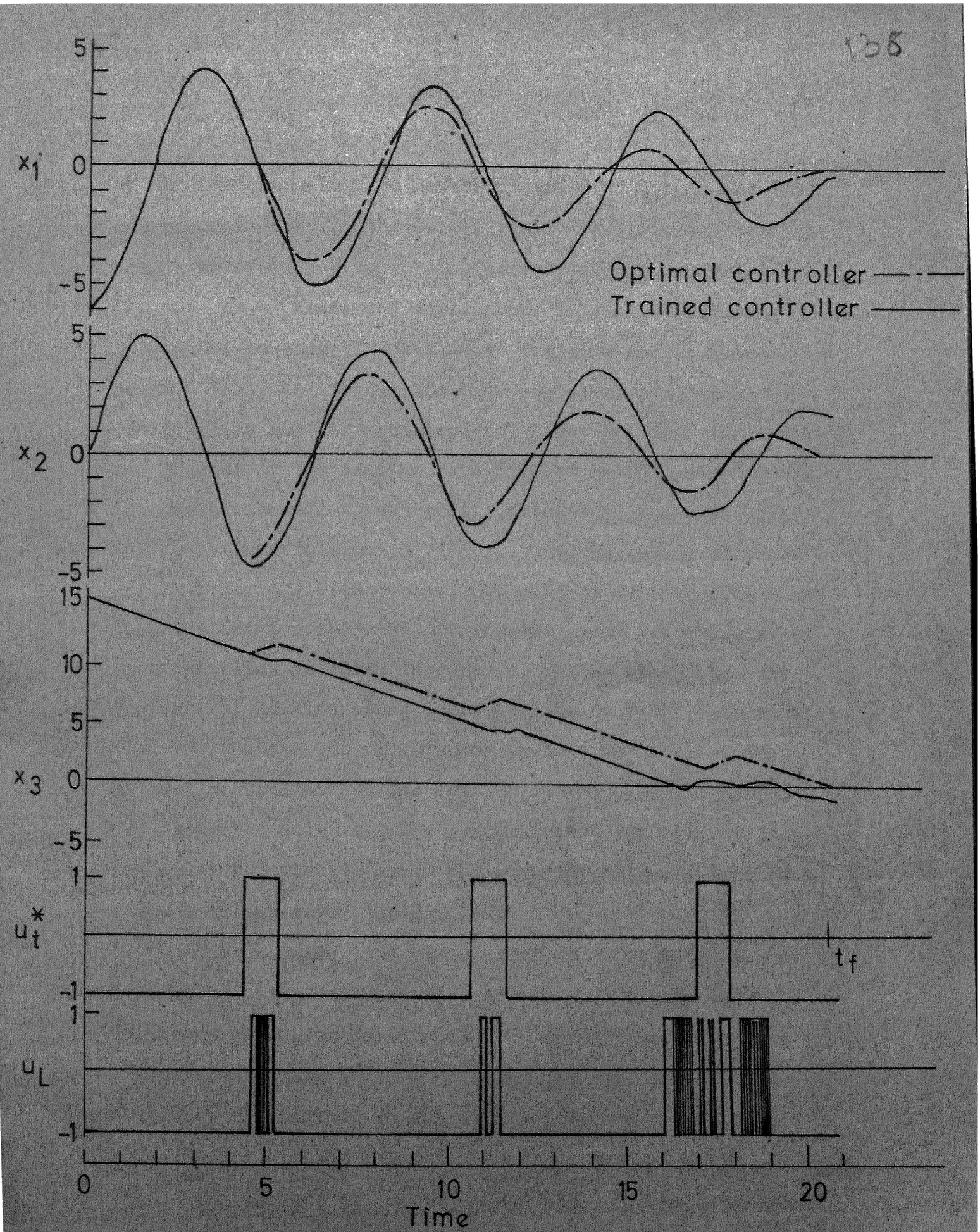
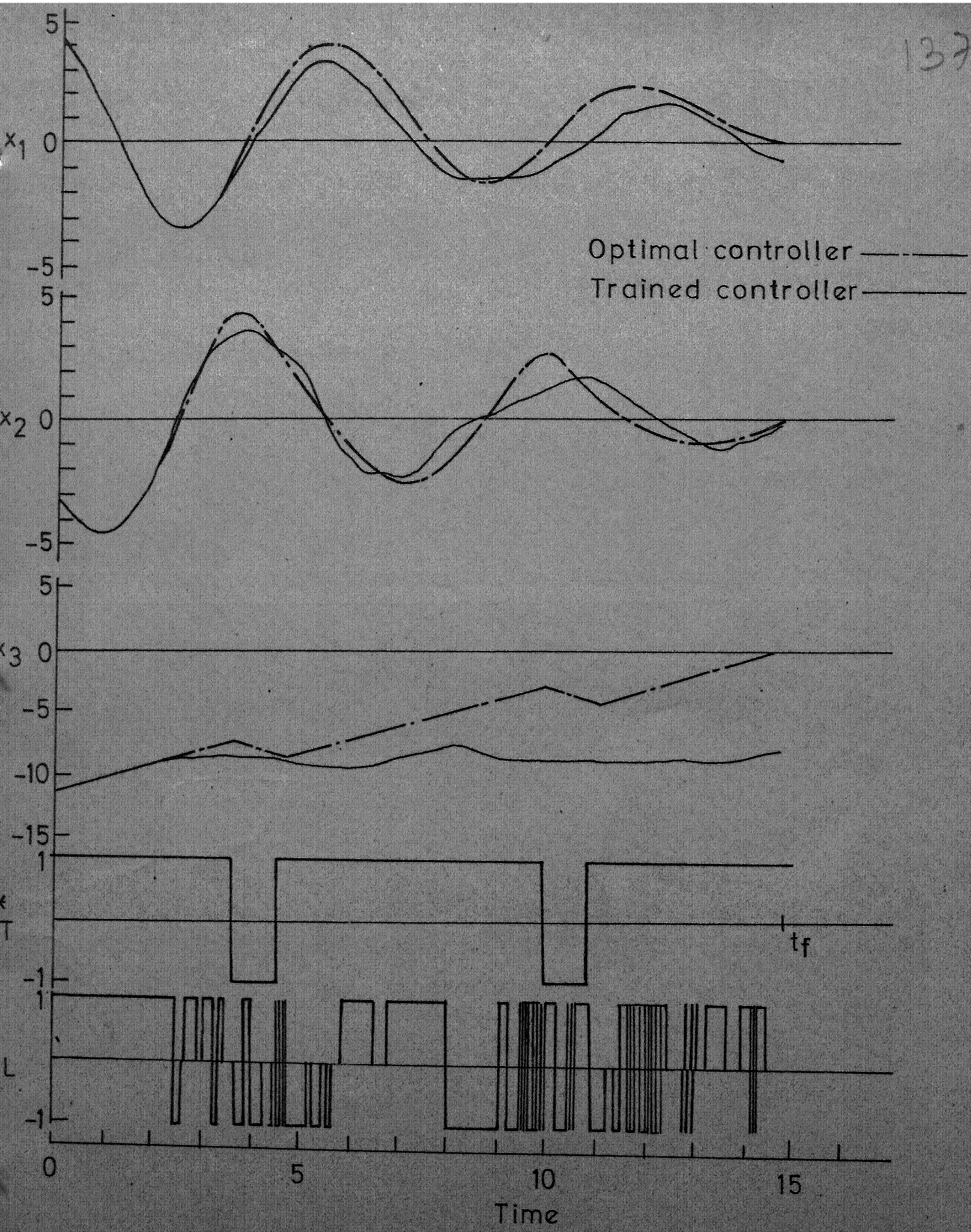


FIG. 3.42 TIME RESPONSE OF TRAINED CONTROLLER WITH ±1 CONTRAST CODE. $\underline{X}(0) = [-6.0, 0.0, 15.0]^T$

of the switching surface in the vicinity of origin and this can be eliminated by the use of finer quantization near the origin or by switching to a different mode of control.

As we mentioned earlier our idea of introducing redundancy in codes is similar to the function of neurons in brain. Tests on trained Adaline were carried out for realiability for different codes. After training was complete, the weights of the Adaline were reduced by 20% using integer arithmetic for all three codes. After this the controller was put to generalization test as shown in Fig. 3.43, 3.44 and 3.45 for different codes. It will be noticed from these figures that for the case of contrast codes the response of the controller is quite favourable, on the other hand the response of Adaline using identity code is badly deteriorated. Table VIII shows the performance in response time of the optimal controller and the trained controller with normal and disturbed weights. The response time for trained controller is the time to drive the states within one quantum control situation of the origin.

From Table VIII it is clear that the Adaline controller using contrast type codes forms a very reliable structure as was expected. We have already examined that the contrast codes yield a powerful means of obtaining excellent generalization properties. Moreover the stored



G.3.43 TIME RESPONSE OF TRAINED CONTROLLER WITH IDENTITY CODE. DISTURBED WEIGHTS. $\underline{X}(0) = [4{\cdot}1, -3{\cdot}2, -11{\cdot}0]^T$

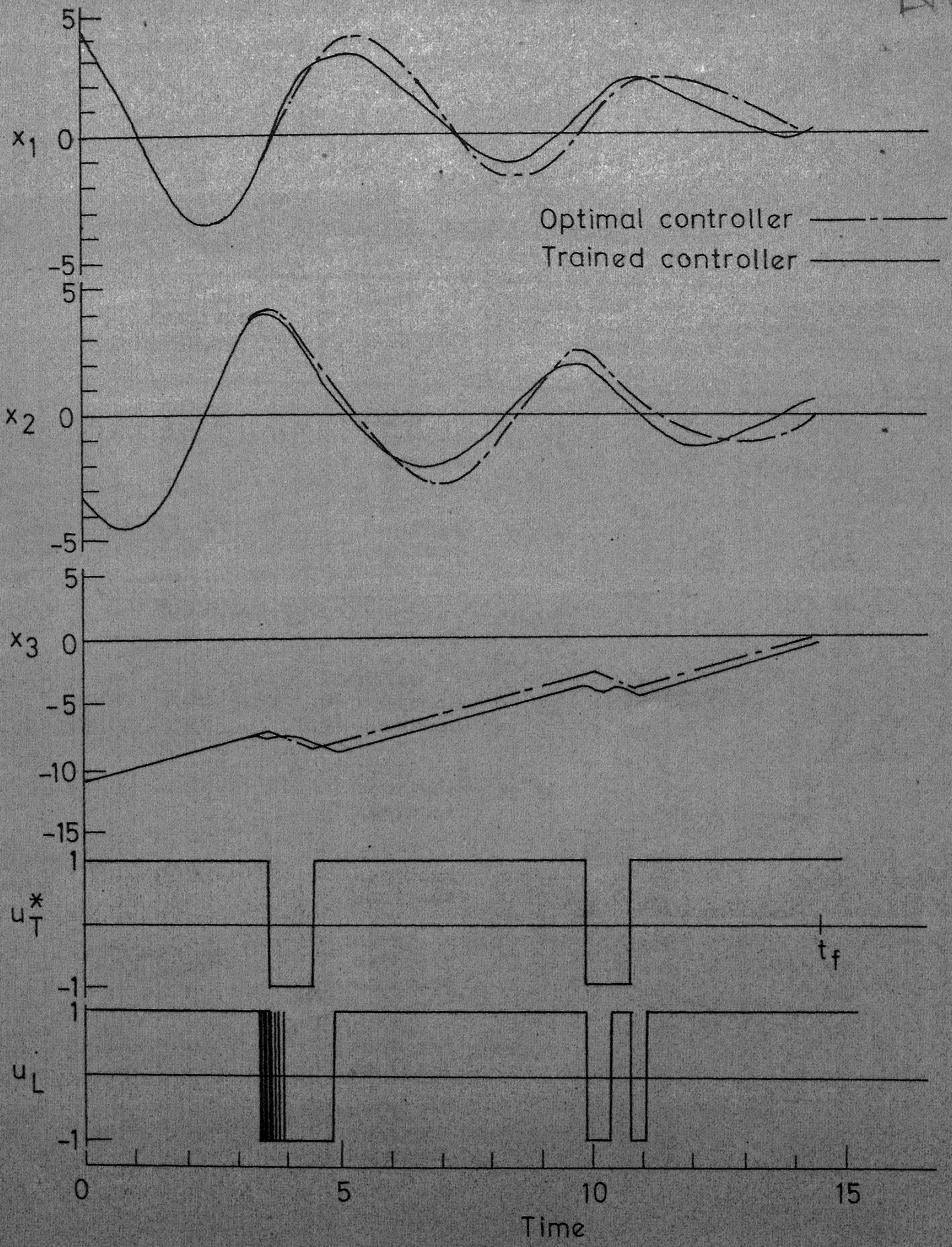


FIG. 3.45 TIME RESPONSE OF TRAINED CONTROLLER WITH ±1 CONTRAST CODE · DISTURBED WEIGHTS.
$\underline{X}(0) = [4.1, -3.2, -11.0]^T$

Table VIII

Actual and optimal response times

Initial condition $\underline{X}(0)$	Code	Optimal time	Trained controller response time	Trained controller response time with 20% disturbed weights
$(-6.0,0.0,15.0)^T$	Identity	20.34	19.94	19.78
	01 contrast		22.28	21.53
	± 1 contrast		23.09	22.39
$(5.0,3.0,0.0)^T$	Identity	10.22	16.73	21.53
	01 contrast		7.29	7.85
	± 1 contrast		8.01	7.85
$(-5.0,0.0,-12.5)^T$	Identity	18.04	23.42	23.38
	01 contrast		17.12	17.14
	± 1 contrast		17.05	16.90
$(4.1,3.2,11.0)^T$	Identity	17.63	21.33	20.66
	01 contrast		16.81	18.94
	± 1 contrast		20.57	20.37
$(4.1,3.2,-11.0)^T$	Identity	15.36	21.71	23.03
	01 contrast		14.54	14.50
	± 1 contrast		14.76	14.17
$(5.0,0.0,0.0)^T$	Identity	8.16	14.27	14.01
	01 contrast		7.34	7.85
	± 1 contrast		7.03	7.08
$(5.35,3.86,-14.32)^T$	Identity	20.83	28.37	-
	01 contrast		20.64	20.77
	± 1 contrast		19.96	19.68
$(5.0,-3.0,12.5)^T$	Identity	18.45	20.60	25.75
	01 contrast		18.70	19.12
	± 1 contrast		19.72	19.08
$(6.0,0.0,-15.0)^T$	Identity	20.34	36.61	-
	01 contrast		19.91	20.05
	± 1 contrast		19.58	19.35

Continued....

Table VIII continued

$(4.1,-3.2,-11.0)^T$	Identity	14.62	17.68	24.49
	01 contrast		15.96	14.11
	± 1 contrast		14.18	13.70
$(5.0,-3.0,-12.5)^T$	Identity	18.67	18.93	25.66
	01 contrast		19.05	19.03
	± 1 contrast		17.68	17.64

information is less exposed to variation in weights in case of contrast codes, since information during training is diffused in a large number of weights.

The most important conclusion from the above work is that the learning controllers can achieve a practical realization of the complete closed loop suboptimal control law with a small set of training samples. It is termed practical for the following reasons.

(1) Quantization need not be very fine

(2) The size of training set can be small

(3) Controller response is less affected by changes in weights

(4) Excessive amount of data processing time is not required

(5) Hardware implementation using chemical cells, integrated circuits etc. is straightforward

CHAPTER 4

CONCLUSIONS AND SCOPE OF FURTHER WORK

In the earlier chapters, we have shown that the application of artificial intelligence techniques are an effective means of realizing arbitrary optimal switching surfaces. The advantages of trainable controllers are (i) the synthesis of the closed loop optimal controller is accomplished simultaneously with training, (ii) the trained controller, because of its special structure, has a higher realiability to component failure than other more conventional designs and (iii) the trained controller provides a complete definition of the closed loop control law from only a small training set.

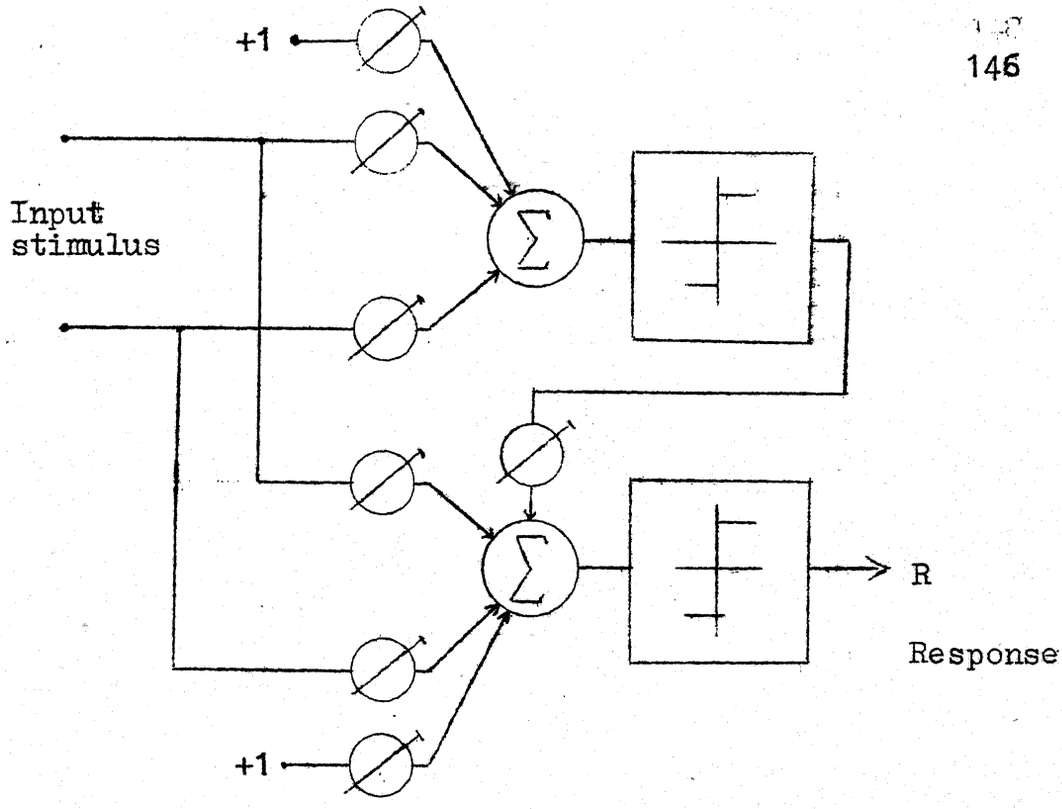
We have already seen that the choice of code affects the generalization characteristics and the convergence rate of learning process to a large extent. It was also pointed out that the controller using a contrast type code is more reliable in case of changes in weights. It has been the experience that the contrast type coded controller always exhibits excellent generalization characteristics. We had also shown that the algorithm of weight correction also affects the convergence rate to a great extent and there exists an optimum mode of weight correction for a given training set.

These experiences indicate that it is not necessary to impose too stringent design requirements on the performance of the components of a learning machine, i.e. a greater tolerance value is permissible for operating magnitude and reliability of a component can be low. This specially makes the use of such machines more attractive in a hazardous and hostile environment.

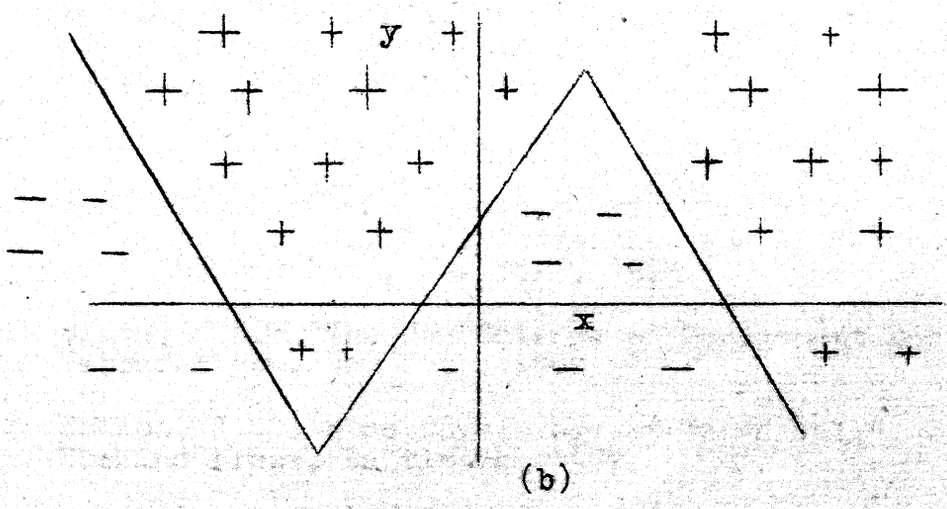
We have also suggested a method for realizing fuel optimal control law by using a network of two Adalines in parallel. It is easy to demonstrate that for n-level of input control a minimum of n-1 Adalines is needed, along with an (n-1) bit decoder. In the previous chapter in case of fuel optimal learning control systems a particular control level (+1, 0 or -1) was observed to be identified if there exists a plane which separates the members of that class from all other classes. This suggests that for n-control levels, a minimum of (n-1) Adalines in parallel would be needed. It will be noticed that in method suggested by Mendel [29], the training of the various Adalines tends to be independent of each other, and this usually leads to a slow convergence of the training process (This also has been pointed out by Nilsson[26]). The alternative method suggested here partially eliminates this. By the use of the code of Table IV, the training of Adalines is not

independent, and it has been found that this method leads to a faster convergence.

The learning machines which we studied here were linear machines. An instance may occur where a linear machine fails to classify all the training patterns correctly (i.e. a nonprojectable dichotomy). In such a case it is desirable to use a cascaded network of several Adalines. Fig. 4.1.3 shows a typical network of Adalines and the form of dichotomy realized by this network. A system of such Adalines is called a layered machine in which the outputs of a layer of Adalines constitute input stimulus to another layer of Adalines and so on. A layered machine provides flexibility of processing the input stimulus before it is finally classified. Unfortunately little is known about the full potential about these machines (i.e. efficient training algorithms are not known and no formulae yet exist to calculate their capacity). Detailed discussion in this field can be found in Rosenblatt [31], Nilsson [26], Cadzow [32] and Holdermann [33] and Block [34].



(a)



(b)

Fig. 4.4 (a) A cascaded network of Adalines, and (b) the dichotomy realized.

REFERENCES

1. Bellman, R.: "Dynamic Programming", Princeton University Press, Princeton, N.J., 1957.

2. Pontryagin, L.S., V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko: "The Mathematical Theory of Optimal Processes", Interscience Publishers, Inc., New York, 1962.

3. Neustadt, L.W.: "Synthesizing Time Optimal Control Systems", Journal of Mathematical Analysis and Applications, Vol. 1, pp. 484-493, 1960.

4. Athans, M. and Falb, P.L.: "Optimal Control", McGraw Hill Book Co., New York, 1966.

5. Plant, J.B.: "Some Iterative Solutions in Optimal Control", M.I.T. Press, 1968.

6. Kirk, Donald E.: "Optimal Control Theory", Englewood Cliffs, N.J., Prentice Hall Inc., 1970.

7. Pavlov, V.I.: "Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebellar Cortex", Dover Inc., N.Y., 1960.

8. Random House Dictionary of English Language, 1970.

9. Turing, A.M.: "Can Machines Think?", in Computers and Thought, 5th ed., E.A. Feigenbaum and J. Feldman, eds., McGraw Hill, New York, 1963.

10. Simon, H.A.: "The New Science of Management Decision", Harper & Row, New York, 1960.

11. Eccles, J.C.: "The Physiology of Nerve Cell", John Hopkins Press, Baltimore, 1957.

12. Shannon, C.E.: "A Symbolic Analysis of Relay and Switching Circuits", Trans. AIEE 57, pp. 713-723, 1938.

13. McCulloch, W.S., Pitts, W.: "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, No. 5, pp. 115-133, 1943.

14. Ashby, W. Ross: "Design for a Brain", Chapman & Hall, London, 1966.

15. Hebb, D.O.: "The Organization of Behaviour", John Wiley & Sons Inc., New York, 1949.

16. Hughes, G.H.: "On the Mean Accuracy of Statistical Pattern Recognizers", IEEE Transactions on Information Theory, IT-14, pp. 55-63, January 1968.

17. Widrow, B.: "Generalization and Information Storage in Networks of Adaline Neurons", in Self organizing Systems – 1962, M.C. Youitts, G.T. Jacobs and G.D. Goldstein, eds., Spartan, New York, 1962.

18. Berkovec, J.W. and Epley, D.L.: "On Time Optimal Control with Threshold Logic Units", 1964 WESCON, August 1964.

19. Winder, R.O.: "Enumeration of Seven Argument Threshold Functions", IEEE Transactions on Electronic Computers, EC-14, No. 1, pp. 315-325, June 1965.

20. Smith Jr., F.B.: "A Logical Net Mechanization for Time Optimal Regulation", NASA Tech. Note TN D-1678, 1962.

21. Smith, F.W.: "A Trainable Nonlinear Function Generator", IEEE Transactions on Automatic Control, AC-11, No. 2, pp. 212-218, 1966.

22. Mendel, J.M. and Zaplac, J.J.: "The Application of Techniques of Artificial Intelligence to Control System Design", in Advances in Control Systems, Vol. 6, C.T. Leondes ed., Academic Press, New York, 1968.

23. Mendel, J.M., Harding, C.F. and Byrne, W.F.: "Feasibility of Realizing a Pattern Recognition Model for Man in the Loop", in Self Organizing Control Systems, Vol. 6, Rept. No. DAC-60603, Douglas Aircraft Co., Santa Monica, Calif., 1967.

24. Smith, F.W.: "Contactor Control by Adaptive Pattern Recognition Techniques", Rept. No. 6762-1, Stanford Electron Lab., Stanford University, Palo Alto, Calif., 1964.

25. Duda, R.C. and Hart, P.E.: "Pattern Classification and Scene Analysis", J. Wiley & Sons Inc., New York, 1973.

26. Nilsson, N.J.: "Learning Machines", Foundations of Trainable Pattern Classifying Systems", McGraw Hill Inc., New York, 1965.

27. Motzkin, T.S. and Schoenberg, I.J.: "The Relaxation Method for Linear Inequalities", Canadian Journal of Mathematics, No. 6, pp. 393-404, 1954.

28. Isenhour, T.L. and Jurs, P.C.: "Some Chemical Applications of Machine Intelligence", Journal of Analytical Chemistry, Vol. 43, August 1971, pp. 20A-35A.

29. Mendel, J.M. and Fu, K.S.: "Adaptive, Learning and Pattern Recognition Systems, Theory and Applications", Academic Press, New York, 1970.

30. Mendel, J.M.: "Self Organizing Control Systems", Vol. 2, Douglas Report SM-47904, Douglas Aircraft Co., Santa Monica, Calif., 1965.

31. Rosenblatt, F.: "Principles of Neurodynamics", Spartan Book Co. Inc., Washington D.C., 1962.

32. Cadzow, J.A.: "Synthesis of Nonlinear Decision Boundaries by Cascaded Threshold Gates", IEEE Transactions on Computers C-17, No. 12, pp. 1165-1172, Dec. 1968.

33. Holdermann, F.: "Classification by Cascaded Threshold Elements", Pattern Recognition, Vol. 3, pp. 243-251. 1971.

34. Block, H.D.: "The Perceptron: A Model for Brain Functioning. I" and "Analysis of a four layer series coupled Perceptron. II", Reviews of Modern Physics, Vol. 34, No. 1, Jan. 1962.

Date Slip

This book is to be returned on the date last stamped.

CD 6.72.9